JRC Scientific and Technical Reports

SOAP Primer for INSPIRE Discovery and View Services

Matteo Villa, Giovanni Di Matteo
TXT e-solutions

Roberto Lucchi, Michel Millot, Ioannis Kanellopoulos
European Commission
Joint Research Centre
Institute for Environment and Sustainability

EUR 23704 EN

The mission of the Institute for Environment and Sustainability is to provide scientific-technical support to the European Union's Policies for the protection and sustainable development of the European and global environment.

European Commission
Joint Research Centre
Institute for Environment and Sustainability

Contact information

Matteo Villa
TXT e-Solutions S.p.A.
Via Frigia, 27
20126 Milano (MI)
ITALY
E-mail: matteo.villa@txt.it
Tel.: +39 02 25771
Fax: +39 02 25771828

Giovanni Di Matteo
TXT e-Solutions S.p.A.
Via Frigia, 27
20126 Milano (MI)
ITALY
E-mail: giovanni.dimatteo@txt.it
Tel.: +39 02 25771
Fax: +39 02 25771828

Roberto Lucchi
European Commission Joint Research Centre
Institute for Environment and Sustainability
Spatial Data Infrastructures Unit
TP262, via Fermi 2749
I-21027 Ispra (VA)
ITALY
E-mail: Roberto.Lucchi@jrc.it
Tel.: +39 0332 78 5325
Fax: +39 0332 78 6325

Michel Millot
European Commission Joint Research Centre
Institute for Environment and Sustainability
Spatial Data Infrastructures Unit
TP262, via Fermi 2749
I-21027 Ispra (VA)
ITALY
E-mail: Michel.Millot@jrc.it
Tel.: +39 0332 78 6146
Fax: +39 0332 78 6325

Ioannis Kanellopoulos
European Commission Joint Research Centre
Institute for Environment and Sustainability
Spatial Data Infrastructures Unit
TP262, via Fermi 2749
I-21027 Ispra (VA)
ITALY
E-mail: Ioannis.Kanellopoulos@jrc.it
Tel.: +39 0332 78 5115
Fax: +39 0332 78 6325

http://ies.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

Legal Notice
Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of this publication.

Europe Direct is a service to help you find answers to your questions about the European Union

Freephone number (*):

00 800 6 7 8 9 10 11

(*) Certain mobile telephone operators do not allow access to 00 800 numbers or these calls may be billed.

A great deal of additional information on the European Union is available on the Internet.
It can be accessed through the Europa server http://europa.eu/

JRC 49175
EUR 23704 EN
ISBN 978-92-79-11317-8
ISSN 1018-5593
DOI 10.2788/78376

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

s

Table of Contents

Table of Figures

1. Introduction

This document demonstrates the use of the proposed INSPIRE SOAP Framework for the INSPIRE Discovery and View services. This document focuses on the analysis of the WSDL itself (for both the Discovery and View services), explaining its parts and characteristics, as well as on the analysis of SOAP request and response messages, including headers and potential attachments. Moreover, the primer is providing also examples of user scenarios, with specific code samples.

2. Purpose and Scope of the Document

The goal of this document is to provide a first release of the INSPIRE SOAP binding primer for *Discovery* and *View* services.

This document is structured in the following way:

- **Chapter 1** is the introduction.
- **Chapter 2**, this one, explains the purpose and the morphology of the document.
- **Chapter 3** provides normative references.
- **Chapter 4** describes and analyses the WSDL of INSPIRE *Discovery* and *View* services, basing mainly on [EUR 23635 - 2008], giving particular attention to the common technical aspects treated in the SOAP framework, supplying useful code sample for each topic.
- **Chapter 5** describes and analyses the WSDL of INSPIRE *Discovery* and *View* services independently, giving care to the single methods WSDL pieces and comparing them with the corresponding OGC implementations.
- **Chapter 6** makes a straightforward analysis about the compliance of the generated WSDLs with the INSPIRE SOAP framework, as defined in [EUR 23635 - 2008].
- **Chapter 7** houses the conclusions derived from this study.

3. Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

REF	AUTHORS	TITLE	YEAR
INSPIRE TechG Disc/1.0	NSDT	Draft Technical Guidance Document for INSPIRE Discovery Services http://inspire.jrc.ec.europa.eu/implementingRulesDocs_ns.cfm	2008
INSPIRE TechG View/0.2	NSDT	Draft Technical Guidance to implement INSPIRE View Services http://inspire.jrc.ec.europa.eu/implementingRulesDocs_ns.cfm	2008
MTOM	Gudgin M., Mendelsohn N. et al.	SOAP Message Transmission Optimization Mechanism http://www.w3.org/	2005
SOAP/1.1	Box D., Ehnebuske D. et al.	Simple Object Access Protocol (SOAP) 1.1 http://www.w3.org/	2000
SOAP/1.1-MTOM	Angelov D., Karmarkar A. et al.	SOAP 1.1 Binding for MTOM 1.0 http://www.w3.org/	2006
WSDL Disc-View	Villa M., Di Matteo G. et al.	Final release of the WSDL for INSPIRE View and Discovery network services http://inspire.jrc.ec.europa.eu/implementingRulesDocs_ns.cfm	2008
EUR 23635 - 2008	Villa M., Di Matteo G. et al.	INSPIRE Network Services SOAP framework http://inspire.jrc.ec.europa.eu/implementingRulesDocs_ns.cfm	2008
WSDL/1.1	Christensen E., Curbera F. et al.	Web Services Description Language (WSDL) 1.1 http://www.w3.org/	2001

Table 1 - Normative References

4. SOAP messages and WSDL for INSPIRE Discovery and View Service

In this chapter we examine all the common technical aspects of the INSPIRE services WSDL, analysing its single sections with reference to the WSDL specification and supplying fitting examples taken from Discovery and View services WSDL.

4.1 Target Namespaces

In INSPIRE WSDLs there will be the need to define target namespace URIs for each one of them, so that their elements will be distinguishable from similar names in different WSDL namespaces.

The chosen URIs won't affect the goodness of the created WSDLs, so their values are not critical and will be just a proposal for the Drafting Team.

4.1.1 INSPIRE example

As we can see from the following snippet of code, the target namespace (highlighted in bold) will have the same prefix for all services ("http://inspire.jrc.ec.europa.eu/") and the suffix depending from the service name:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
  targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soapEnv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:insp="http://inspire.jrc.ec.europa.eu/">

   …

</wsdl:definitions>
```

Example 4.1.1-1 Discovery Service targetNamespace definition

Other namespaces from OGC and from the used standards are referenced too in the *definition* main element; in the above example we can for instance see *mime*, *csw*, *ogc*, etc.

Common data structures, as headers or common exceptions, will have as namespace the base prefix URL "http://inspire.jrc.ec.europa.eu/".

4.2 Types

As in all languages in which there is an exchange of data, even in WSDL it is essential to begin a document with the definition of the data-types available for the use.
We will use XML Schema to do so, as talked on in [EUR 23635 - 2008], when we chose to use the *Document-literal wrapped* encoding style.
WSDL allows message types to be defined directly inside the WSDL document, housed in the *types* element, or to use the *import* mechanism, providing type definitions in separate documents.

Data-types common to both INSPIRE View and Discovery services are grouped together in a separated XSD file to be easily imported in both the WSDLs and maybe re-used by the other INSPIRE services too.

4.2.1 INSPIRE example

In INSPIRE WSDLs we use both the methods supplied by WSDL to define message types.

We import useful XSD schemas provided by OGC, like in the following example, taken from *Discovery Service* WSDL, where CSW and common OWS data definition schema are imported:

```
<wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
      <!-- import all the necessary XML schema --!>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsServiceIdentification.xsd"/>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsServiceProvider.xsd"/>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsOperationsMetadata.xsd"/>
      <s:import namespace="http://www.opengis.net/cat/csw/2.0.2"
      schemaLocation="http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd"/>
      <s:import namespace="http://www.opengis.net/cat/csw/2.0.2"
      schemaLocation="http://schemas.opengis.net/csw/2.0.2/CSW-publication.xsd"/>
      <s:import namespace="http://www.opengis.net/cat/csw/2.0.2"
      schemaLocation="http://schemas.opengis.net/csw/2.0.2/record.xsd"/>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsAll.xsd"/>
      <s:import namespace="http://www.opengis.net/ogc"
      schemaLocation="http://schemas.opengis.net/filter/1.1.0/filter.xsd"/>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/ows19115subset.xsd"/>
      <s:import namespace="http://www.opengis.net/ows/1.1"
      schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsExceptionReport.xsd"/>
      <s:import namespace="http://schemas.xmlsoap.org/soap/envelope/"
      schemaLocation="http://schemas.xmlsoap.org/soap/envelope/"/>
```

```
        <!--Inclusion of headers datatypes and elements -->
        <s:import namespace="http://inspire.jrc.ec.europa.eu/" schemaLocation="INSPIRECommon.xsd"/>
    ...
</wsdl:types>
```

Example 4.2.1-1 Discovery Service XSD schema import

The last import showed in the previous example includes the INSPIRE data-types common to all services.

But we also define new message types to be used in the WSDL, whose elements can anyway be referenced from OGC external XSDs:

```
<wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
      ...
      <s:element name="GetDiscoveryServiceMetadataResponse">
        <s:complexType>
          <s:sequence>
            <s:element ref="ows:ServiceIdentification" />
            <s:element ref="ows:ServiceProvider" />
            <s:element ref="ogc:Filter_Capabilities" />
            <s:element ref="ows:OperationsMetadata" />
          </s:sequence>
        </s:complexType>
      </s:element>
      ...
  </s:schema>
</wsdl:types>
```

Example 4.2.1-2 Discovery Service new Message type definition

4.3 Messages

Messages are the pieces of information exchanged with the service and are composed by one or more logical parts; each part is associated with a type. Their composition is quite straightforward.

Messages common to both INSPIRE View and Discovery services are grouped together in a separated WSDL file to be imported in both the WSDLs.

4.3.1 INSPIRE example

As a convention, we decided to give names to the single INSPIRE SOAP framework input/output messages following a simple rule: the message name will be composed adding to the name of the service the suffixes “SoapIn” or “SoapOut”, depending on the nature of the message.
On the other hand, the names of the messages bound to the SOAP headers parts will be composed by the name of the service concatenated to the name of the header.

The message used for reporting exception is reported in paragraph *4.9 Exception management*.

Here after we report a sample message taken from the *Discovery Service* WSDL; its structure is as easy as it seems:

```
<wsdl:message name="GetDiscoveryServiceMetadataSoapIn">
    <wsdl:part name="parameters" element="tns:getCapabilities" />
</wsdl:message>
```

Example 4.3.1-1 Discovery Service message example

4.4 Port Types

As defined in [WSDL/1.1], a *Port type* in a WSDL is a named set of abstract operations and the abstract messages involved.
Each INSPIRE service will have its own *Port type* containing all the abstract methods described in its specification. All INSPIRE operations described in this document follow the *Request-response* transmission primitive (for a clear understanding of the different Message Exchange Patterns, we recommend to give a look at section 2.4 of [WSDL/1.1]).

4.4.1 INSPIRE example

As example of *Port type* for an INSPIRE service, we quote here after the one defined in the *Discovery Service* WSDL, where all the five INSPIRE operations are reported:

```
<wsdl:portType name="DiscoveryServiceSoap">
    <wsdl:operation name="GetDiscoveryServiceMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Request to a Service to perform the GetCapabilities operation. This operation allows a client to retrieve a Capabilities XML document providing metadata for the specific INSPIRE server.</wsdl:documentation>
      <wsdl:input message="tns:GetDiscoveryServiceMetadataSoapIn" />
      <wsdl:output message="tns:GetDiscoveryServiceMetadataSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="DiscoverMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">DiscoverMetadata service description.</wsdl:documentation>
      <wsdl:input message="tns:DiscoverMetadataSoapIn" />
      <wsdl:output message="tns:DiscoverMetadataSoapOut" />
      <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
    </wsdl:operation>
    <wsdl:operation name="GetMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The GetMetadata operation allows to retrieve metadata for specific resources from a result set based on the resource unique Identification (ID).</wsdl:documentation>
      <wsdl:input message="tns:GetMetadataSoapIn" />
      <wsdl:output message="tns:GetMetadataSoapOut" />
      <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
    </wsdl:operation>
    <wsdl:operation name="PublishMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The Publish Metadata operation allows to create, delete, or update (set) metadata (record) elements of spatial resources in the Discovery Service datastore.</wsdl:documentation>
      <wsdl:input message="tns:PublishMetadataSoapIn" />
      <wsdl:output message="tns:PublishMetadataSoapOut" />
      <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
    </wsdl:operation>
    <wsdl:operation name="CollectMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The Collect Metadata operation allows to pull metadata (record) elements of spatial resources from a source Discovery Service datastore and allows to create, delete or update (set) the metadata (record) elements of these spatial resources in the target Discovery Service datastore (pull metadata mechanism).</wsdl:documentation>
      <wsdl:input message="tns:CollectMetadataSoapIn" />
      <wsdl:output message="tns:CollectMetadataSoapOut" />
```

```
        <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
      </wsdl:operation>
</wsdl:portType>
```

Example 4.4.1-1 Discovery Service portType section

4.5 Bindings

Up to now we have shown the WSDL parts to specify what kind of abstract messages can be exchanged in the WSDL, but we have not yet specified how those messages can be exchanged. This is the purpose of the *binding*. A *binding* specifies concrete message format and transmission protocol details for a *Port type* and must supply such details for every operation and fault in the *Port type*.

4.5.1 INSPIRE example

Following WSDL specification, there can be more bindings for a given port type, but in INSPIRE case, only a SOAP 1.1 binding is foreseen.

Here after we quote the *Discovery Service* binding for SOAP 1.1:

```
<wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
    <wsdl:documentation>
      <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
        xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
    </wsdl:documentation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetDiscoveryServiceMetadata">
      <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/GetDiscoveryServiceMetadata"
       style="document" />
      <wsdl:input>
        <soap:header message="tns:GetServiceMetadataAuthHeader" part="AuthHeader" use="literal"/>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
        <soap:body use="literal" name="ServiceExceptionReport" />
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="DiscoverMetadata">
      <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/DiscoverMetadata"
      style="document" />
      <wsdl:input>
        <soap:header message="tns:GetServiceMetadataAuthHeader" part="AuthHeader" use="literal"/>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
        <soap:body use="literal" name="ServiceExceptionReport" />
```

```
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="GetMetadata">
        <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/GetMetadata" style="document"
          />
          <wsdl:input>
            <soap:header message="tns:GetServiceMetadataAuthHeader" part="AuthHeader" use="literal"/>
            <soap:body use="literal" />
          </wsdl:input>
          <wsdl:output>
            <soap:body use="literal" />
          </wsdl:output>
          <wsdl:fault name="ServiceExceptionReport">
            <soap:body use="literal" name="ServiceExceptionReport" />
          </wsdl:fault>
        </wsdl:operation>
      <wsdl:operation name="PublishMetadata">
        <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/PublishMetadata"
          style="document"/>
        <wsdl:input>
          <soap:header message="tns:GetServiceMetadataAuthHeader" part="AuthHeader" use="literal"/>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
        <wsdl:fault name="ServiceExceptionReport">
          <soap:body use="literal" name="ServiceExceptionReport" />
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="CollectMetadata">
        <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/CollectMetadata"
        style="document"/>
        <wsdl:input>
          <soap:header message="tns:GetDiscoveryServiceMetadataAuthHeader" part="AuthHeader"
            use="literal"/>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
        <wsdl:fault name="ServiceExceptionReport">
          <soap:body use="literal" name="ServiceExceptionReport" />
        </wsdl:fault>
      </wsdl:operation>
</wsdl:binding>
```

Example 4.5.1-1 Discovery Service SOAP 1.1 binding

Message exchange protocols different from SOAP are clearly not in the scope of this document, but, following OGC recommendation, it could be useful to provide at least a *getCapabilities* web method apart, to make the capabilities document available through the GET protocol.
In the following code piece, it is shown a GET binding for such an operation that wants to be just an example of how to refer such a binding in the service WSDL.

```
<wsdl:portType name="DiscoveryServiceHttpGET">
  <wsdl:operation name="GetServiceMetadata">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Request to a Service to perform the GetCapabilities operation. This operation allows a client to retrieve a Capabilities XML document providing metadata for the specific INSPIRE server.</wsdl:documentation>
    <wsdl:input message="tns:GetServiceMetadataHttpIn" />
    <wsdl:output message="tns:GetServiceMetadataHttpOut" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="DiscoveryServiceHTTPGETBinding" type="tns:DiscoveryServiceHttpGET">
  <http:binding verb="GET"/>
  <wsdl:operation name="GetServiceMetadata">
    <http:operation location="GetServiceMetadata"/>
    <wsdl:input>
      <http:urlEncoded/>
    </wsdl:input>
    <wsdl:output>
      <mime:content type="text/xml"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Listing 1 – Sample HTTP GET binding

In the *DiscoveryServiceHttpGET PortType*, only *GetServiceMetadata* operation is defined. The method input parameters could be redefined to allow an easier parameter encoding in URL and the same assertion can be done for the capabilities document output format.
In the previous sample the input parameter are specified as *urlEncoded*, to simulate the most common GET request, while the output is foreseen as an xml text.

4.6 Ports

As specified in [WSDL/1.1], a *port* defines an individual endpoint by specifying a single address for a binding.

4.6.1 INSPIRE example

In INSPIRE WSDLs, each service has one *port*. Here after there's a snippet code taken from the *Discovery Service* WSDL:

```
<wsdl:port name="DiscoveryServiceSoap" binding="tns:DiscoveryServiceSoap">
  <soap:address location="$SERVICE_URL" />
</wsdl:port>
```

Example 4.6.1-1 Discovery Service SOAP 1.1 port

The *$SERVICE_URL* string represents the URI where the service is publicly accessible.

A specific *Port* for the hypothetical *DiscoveryServiceHTTPGETBinding* binding (see paragraph 4.5.1) should be added if this binding is declared.

4.7 Defining a service

A *service* element simply groups together a set of *port*s referring to the same service.

4.7.1 INSPIRE example

INSPIRE services, up to [WSDL Disc-View], will have just one *port* inside their service element, as demonstrated in the following code snippet taken from *Discovery Service*:

```
<wsdl:service name="DiscoveryService">
  <wsdl:port name="DiscoveryServiceSoap" binding="tns:DiscoveryServiceSoap">
    <soap:address location="$SERVICE_URL" />
  </wsdl:port>
</wsdl:service>
```

Example 4.7.1-1 Discovery Service service

If a new *Port* for the HTTP GET binding is created, it should be added to the *service* element.

4.8 Headers

In this set of paragraphs we are going to pass under examination the SOAP headers underlined as useful for the INSPIRE use cases in [EUR 23635 - 2008]. For each one of those, the binding with the *Discovery Service* will be analysed and a SOAP implementation will be potentially proposed. The same binding can be though for each INSPIRE service.

Headers data-types and elements are common to both INSPIRE View and Discovery services and probably they will be common also to the other INSPIRE services, so we decided to group their definition together in a common XSD file to be imported in each INSPIRE service WSDL, to enhance code reusability and to improve possible future modification to header structures.

4.8.1 Security

The security topic results to be of strategic importance for the INSPIRE domain under several aspects. The INSPIRE framework has the need to protect access to reserved data and to recognise the identities of people accessing to them.
As already described in [EUR 23635 - 2008], to exploit header potential can turn out to be a good choice for performance, code clearness and reduction of redundancies.

In this paragraph we are going to suppose a simple authentication mechanism, realised by a couple of textual username and password, just to represent the way to use the SOAP header for this aim.

A class for the header could be as in the following listing:

```
public class AuthHeader : SoapHeader
{
   public string userName;
   public string password;
}
```

Listing 2 - Sample Authentication header C# class

Once the *AuthHeader* is specified as compulsory for a web-method call, any call lacking *AuthHeaders* won't even reach the web method. Calls that do reach it can be authenticated by reading *Credentials*' *userName* and *password* fields and this is possible even if the two parameters are not passed as direct input to the web method. The web method checks the user name and password and fails the call if either is invalid, throwing a *SoapException*, as defined in the *WSDL*:

```
public AuthHeader Credentials;

[SoapHeader("Credentials", Required = true)]
webMethod(params[])
{
   if (!checkCredentials(Credentials.UserName,Credentials.Password))
      throw new SoapException ("Unauthorized", SoapException.ClientFaultCode);
   …
}
```

Listing 3 - Sample C# code specifying the Authentication header as compulsory and checking the user credentials

The whole credentials control is performed server-side as usual; the difference between the two ways of managing this security issue is the way parameters are exchanged between client and server.

Username and password are no more explicitly part of the method input parameters, but they are sent to the server in parallel with the method input, so they can be treated **independently**.

In this way, every Web method has to check user names and passwords.

The *AuthHeader* will be included in the web method request message and will be available at server side.

These are the changes that this header brings in the web service WSDL:

- the definition of the new data type for authentication

```
<s:element name="AuthenticationHeader" type="tns:AuthHeader" />
<s:complexType name="AuthHeader">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="userName" type="s:string" />
    <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
  </s:sequence>
  <s:anyAttribute />
</s:complexType>
```

Listing 4 - AuthenticationHeader data-type definition in WSDL

- the new message part corresponding to the authentication header

```
<wsdl:message name="GetDiscoveryServiceMetadataAuthHeader">
  <wsdl:part name="AuthHeader" element="insp:AuthenticationHeader" />
</wsdl:message>
```

Listing 5 - AuthenticationHeader message part in WSDL

- the new composition of the SOAP request message

```
<wsdl:operation name="GetDiscoveryServiceMetadata">
  <soap:operation soapAction="/http://inspire.jrc.ec.europa.eu/Discovery/GetDiscoveryServiceMetadata"
   style="document"/>
   <wsdl:input>
     <soap:header message="tns:GetDiscoveryServiceMetadataAuthHeader" part="AuthHeader" use="literal"/>
     <soap:body use="literal" />
  </wsdl:input>
  <wsdl:output>
     <soap:body use="literal" />
   </wsdl:output>
  </wsdl:operation>
```

Listing 6 - New SOAP request message structure

This is a sample of a SOAP request containing the *AuthHeader* header:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <AuthHeader xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
      <userName>donald</UserName>
```

```
        <password>Duck</Password>
      </AuthHeader>
    </soap:Header>
    <soap:Body>
      …
    </soap:Body>
</soap:Envelope>
```

Listing 7 - Sample SOAP request message with AuthHeader

The *soap:mustUnderstand* and *soap:role* attributes are not necessaries and can be omitted or set to their default values (respectively "no" and "ultimateReceiver"), because the check for the presence of the *AuthHeader* header will be performed just by the last node of the path and this node will always check the presence of the header.
Up to now, it would be nonsense to check credentials in nodes different from the last one, because they are considered just transmitting/forwarding nodes and they are not supposed to have knowledge of credentials or user permissions management.

Using SOAP headers for transport means the information is passed out-of-band and that it isn't tied to any particular protocol, such as HTTP. But it is still possible to have room for improvement, maybe implementing a specific SOAP extension to get the authorization code out of the Web methods and into a module that examines incoming requests as they arrive and rejects those lacking valid credentials.
Logically, this would be an object that sits in the HTTP pipeline and enjoys access to the SOAP messages entering and leaving.
If the message contains no *AuthHeader*, or if it contains an *AuthHeader* with invalid credentials, the *AuthExtension* would reject it by throwing a *SoapException*. Such in a way, the web methods will no longer have to do any authentication themselves, because they will never see a message that hasn't been already authenticated.
To realise such a job, for instance, a language like C# allows creating a SOAP extension class deriving it from the pre-defined class *SoapExtension*.
It is quite straightforward that username and password, in the just presented example, will be transmitted in clear text in the SOAP Header. In the "real life" some kind of encryption will be needed for sure. Even in this situation it would be possible to create a SOAP extension aiming to encrypt and decrypt SOAP messages.

4.8.1.1 Network Service Drafting Team Proposal

The conceptual approach discussed in the new version of the Network Services architecture ([INSPIRE NSA/3.0]) is similar to the one proposed in this study: the service access management is foreseen as common to all INSPIRE services and credentials providing is solved exploiting a dedicated SOAP header block where an identification token is stored.

Here after we report a simple message equipped with a right management header as described in [INSPIRE NSA/3.0]:

```
<soapenv:Envelope xmlns:soapenv=…>
  <soapenv:Header>
    <inspire:rightsManagement xmlns:inspire="http://..." soapenv:mustUnderstand="1">
      <rightsManagementKey>…</rightsManagementKey>
    </inspire:rightsManagement>
  </soapenv:Header>
```

```
  <soapenv:Body>
    // service call
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 8 - NSDT rightsManagement header message sample

Architectural and low-level user rights management is out of the scope of this study, so it won't be treated in this document.

4.8.2 Checksums & Signatures

As it will be explained in §0, no attachment is foreseen for *Discovery Service*, so no checksum header is required, while in *View Service*, the *GetMap* operation returns an image representing a geo-spatial map, so a checksum should be useful to check the integrity of downloaded data.

The simplest implementation for such functionality would be a class with just one integer (32bit) parameter to store the file checksum.

```
public class ChecksumHeader : SoapHeader
{
   public Int32 checksum;
}
```

Listing 9 - Sample Checksum header C# class

The *ChecksumHeader* will be included in the web method response message and will be available at client side.
These are the changes brought in the web service WSDL:

- the definition of the new element and data type for checksum control

```
<s:element name="ChecksumHeader" type="tns:ChecksumHeader" />
<s:complexType name="ChecksumHeader">
   <s:sequence>
      <s:element name="checksum" type="s:int" />
   </s:sequence>
   <s:anyAttribute />
</s:complexType>
```

Listing 10 - ChecksumHeader element and data-type definition in WSDL

- the new message part corresponding to the human readable information header

```
<wsdl:message name="GetMapChecksumHeader">
   <wsdl:part name="ChecksumHeader" element="insp:ChecksumHeader" />
 </wsdl:message>
```

Listing 11 - ChecksumHeader message part in WSDL

- the new composition of the SOAP response message

```
<wsdl:operation name="GetMap">
    <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/View/GetMap" style="document" />
    <wsdl:input>
       <soap:body use="literal" />
       <soap:header message="tns:GetMapAuthHeader" part="AuthHeader" use="literal" />
    </wsdl:input>
    <wsdl:output>
       <soap:body use="literal" />
       <soap:header message="tns:GetMapChecksumHeader" part="ChecksumHeader" use="literal" />
    </wsdl:output>
</wsdl:operation>
```

Listing 12 - New SOAP request message structure with Checksum header

This is a sample of the SOAP response containing the checksum header:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Header>
  <ChecksumHeader xmlns="http://inspire.jrc.ec.europa.eu/GetMap">
   <checksum>1126792343</checksum>
  </DescriptionHeader>
 </soap:Header>
 <soap:Body>
  …
 </soap:Body>
</soap:Envelope>
```

Listing 13 - Sample SOAP request message

The *soap:mustUnderstand* and *soap:role* attributes can be omitted, because the recipient of the human readable header is the last node of the path and its understanding is not compulsory.

The signature topic, instead, could be useful in the *Discovery Service* context as for other INSPIRE services, allowing the client to be sure to have received valid data from a certified source.
In this perspective INSPIRE is giving a look at *eID* at EC level (*http://ec.europa.eu/information_society/activities/egovernment/policy/key_enablers/eid/index_en.htm*) , to manage the signature problem in a standard way and in the next future it will follow its developments.

4.8.3 Description & Human-readable information

The "description & Human-readable" header is probably the most general one, because it can be exploited to supply any kind of human readable information together with the normal server response. This is the reason why any INSPIRE service can use this header for the just cited purpose.

The simplest implementation for such functionality would be based on a class with just one string parameter containing the human readable text:

```
public class DescriptionHeader : SoapHeader
{
   public string message;
}
```

Listing 14 - Sample Description header C# class

The *DescriptionHeader part* will be included in the web method response message and will be available at client side.
These are the changes brought in the web service WSDL:

- the definition of the new data type for human readable information

```
<s:complexType name="DescriptionHeader">
   <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="message" type="s:string" />
   </s:sequence>
   <s:anyAttribute />
</s:complexType>
```

Listing 15 - DescriptionHeader data-type definition in WSDL

- the new message part corresponding to the human readable information header

```
<wsdl:message name="GetDiscoveryServiceMetadataDescriptionHeader">
   <wsdl:part name="DescriptionHeader" element="insp:DescriptionHeader" />
</wsdl:message>
```

Listing 16 - DescriptionHeader message part in WSDL

- the new composition of the SOAP response message

```
<wsdl:operation name="GetDiscoveryServiceMetadata">
   <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/GetDiscoveryServiceMetadata"
     style="document"/>
    <wsdl:input>
      <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
      <soap:header message="tns:GetDiscoveryServiceMetadataDescriptionHeader" part="DescriptionHeader"
       use="literal" />
      <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
```

Listing 17 - New SOAP request message structure

This is a sample of the SOAP response containing the human readable information header:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Header>
  <DescriptionHeader xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
   <message>this is a sample human readable message stored in the header of the web method
   response</message>
  </DescriptionHeader>
 </soap:Header>
 <soap:Body>
  …
 </soap:Body>
</soap:Envelope>
```

Listing 18 - Sample SOAP request message

The *soap:mustUnderstand* and *soap:role* attributes can be omitted, because the recipient of the human readable header is the last node of the path and its understanding is not compulsory.

4.8.4 Metadata & Piggy-backing - Multilingualism

Looking at its nature and at its use in INSPIRE, multilingualism can be easily considered as a typical metadatum.
As chosen in [EUR 23635 - 2008], the multilingualism aspect will be faced in an apposite header, responsible to manage the requests done by clients about the be-wished language.

The multilingualism aspect needs to be managed by all the INSPIRE services and in all the client requests and this is the main reason because the header section solution is the most appropriate one.

The multilingual header will contain just a parameter to store the ISO standard code of the client be-wished language (as specified in [ISO 639-2/B]) and a class for this header could be as the one in the following listing:

```
public class MultilingualHeader : SoapHeader
{
   public string lang;
}
```

Listing 19 - Sample Multilingual header C# class

The *MultilingualHeader* will not be specified as compulsory for any web-method call, so any call lacking *MultilingualHeader* will reach the web method anyway and the default language will be automatically chosen. Calls bringing the *MultilingualHeader* will allow the web method to check whether the specified language code is valid and in that case they will use that idiom for the response.

```
public MultilingualHeader multilingual;

[SoapHeader("multilingual")]
webMethod(params[])
{
   String lang = manageLanguageCode(multilingual.lang);
   …
}
```

Listing 20 - Sample C# code specifying the Multilingual header as optional and managing the client request

The *MultilingualHeader* will be included in the web method request and will be available at client side.

These are the changes brought in the web service WSDL by the multilingualism header:

- the definition of the new data type for the multilingual choice

```
<s:complexType name="MultilingualHeader">
  <s:sequence>
    <s:element minOccurs="0" maxOccurs="1" name="lang" type="s:string" />
  </s:sequence>
  <s:anyAttribute />
</s:complexType>
```

Listing 21 - MultilingualHeader data-type definition in WSDL

- the new message part corresponding to the multilingual header

```
<wsdl:message name="GetDiscoveryServiceMetadataMultilingualHeader">
  <wsdl:part name="MultilingualHeader" element="insp:MultilingualHeader" />
</wsdl:message>
```

Listing 22 - MultilingualHeader message part in WSDL

- the new composition of the SOAP request message

```
<wsdl:operation name="GetDiscoveryServiceMetadata">
  <soap:operation soapAction=" http://inspire.jrc.ec.europa.eu/Discovery/GetDiscoveryServiceMetadata" style="document"/>
  <wsdl:input>
    <soap:header message="tns:GetDiscoveryServiceMetadataMultilingualHeader" part="MultilingualHeader"
     use="literal" />
    <soap:body use="literal" />
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
```

Listing 23 - New SOAP request message structure

This is a sample of the SOAP request containing the *MultilingualHeader* header:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <MultilingualHeader xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
      <lang>ITA</lang>
    </MultilingualHeader>
  </soap:Header>
  <soap:Body>
    ....
  </soap:Body>
</soap:Envelope>
```

Listing 24 - Sample SOAP request message with MultilingualHeader

The *soap:mustUnderstand* and *soap:role* attributes can be omitted, because the check for the presence of the *MultilingualHeader* header will be performed just by the last node of the path and this node will always use this kind of header to manage multilingualism.

4.9 Exception management

In case an exception is encountered while processing a Discovery service operation request, encoded in a SOAP envelope, the INSPIRE server must generate a SOAP response message where the content of the <*soap:Body*> element is a <*soap:Fault*> element. The following skeleton XML fragment must be used when generating the <*soap:Body*> element in the event that the CSW server encounters an exception:

```
<soap:Body>
  <soap:Fault>
    <faultcode>soap:Server</faultcode>
    <faultstring>A server exception was encountered.<faultstring>
    <detail>
      <ows:ExceptionReport>
        ...
      </ows:ExceptionReport>
    </detail>
  </soap:Fault>
</soap:Body>
```

Listing 25 - SOAP fault response message

The <*faultcode*> element shall have the content "soap:Server" indicating that this is a server exception. The <*faultstring*> element shall have the content "A server exception was encountered.". This fixed string is used since the details of the exception are specified in the <*detail*> element using an <*ows:ExceptionReport*>.

As much as this implementation uses OGC services as lower layer for INSPIRE ones, we suggest that the <*detail*> tag shall contain an <*ows:ExceptionReport*> element detailing the specific exception that the server encountered, as OGC does. It could be useful to underline that this is just a possible proposal for this aspect, bound to our implementation.

The referred XSD schema for <*ows:ExceptionReport*> data-type can be found at http://schemas.opengis.net/ows/1.1.0/owsExceptionReport.xsd.

The exception structure has been inspired from the OGC one, re-adapted to the <*soap:Fault*> element structure, as foreseen by [SOAP/1.1] and WS-I basic profile 1.2.

An ad hoc WSDL message has been created to manage exceptions and it will be used by all methods:

```
<wsdl:message name="ServiceExceptionReport">
  <wsdl:part element="soapEnv:Fault" name="Body"/>
</wsdl:message>
```

Listing 26 - WSDL message to manage exceptions

This message has been defined in a separated WSDL, to be imported by all the INSPIRE services WSDLs, improving code reusability and avoiding different implementation for different services.

For each web method of Discovery and View services the following line will be added in the *portType* to specify the error message to be used in case of fault:

```
<wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport"/>
```

Listing 27 - Common WSDL fault element

In the SOAP binding area of the WSDL, each operation will have the following WSDL code to specify the encoding style to be applied to the fault:

```
<wsdl:fault name="ServiceExceptionReport">
   <soap:body use="literal" name="ServiceExceptionReport" />
</wsdl:fault>
```

Listing 28 - SOAP encoding of Exception messages

4.10 MTOM and XOP use

As already described in [EUR 23635 - 2008], XOP allows the serialization of an XML Infoset containing *base64*-encoded data as a package. As easily illustrated in [Ghosh, 2007], the package is still in an XML form, with the root of the package containing the XML document itself, while selected portions of the package content are *base64*-encoded binary data extracted and re-encoded (i.e., the data is decoded from *base64*) and placed into the package. The locations of those selected portions are marked in the XML with a special element that links to the packaged data using URIs. To be precise, XOP defines how to use MIME multipart/related as the underlying packaging system for a XOP package. Here the *base64*-encoded binary content has been serialized into separate MIME parts. The serialization and deserialisation happens in few steps as shown in the next Figure.

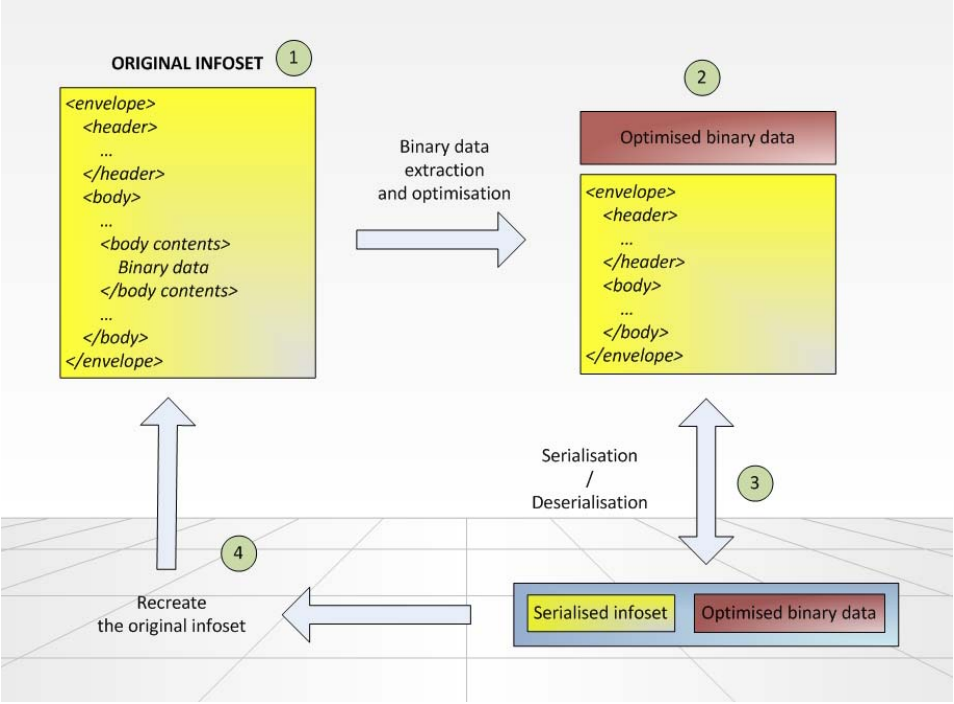

Figure 1 - Steps in sending an MTOM Message (in circles the step number)

First the original XML Infoset is taken and the *base64* encoded binary content is processed. This is marked in Figure 1 by the number 1, circled. Selected portions of its content, that have *base64*-encoded binary data, are extracted and re-encoded (i.e., the data is decoded from *base64*) and placed into the package. This optimized content is removed from the Infoset as depicted in step 2 of the Figure 1. This forms the XOP Infoset. The locations of these selected portions are marked in the XML with a special element that links to the packaged data using URIs. Here the optimized content in the original Infoset will be replaced by the *xop:Include* element. Then the XOP Infoset with the optimized contents will be serialized as shown by step 3 of the Figure 1. Here the binary contents has been serialized in separate MIME parts and will not be part of the XOP Infoset .This forms the XOP package containing the XOP document which in turn comprises the serialized XOP Infoset and any optimized content. The XOP package acts as the alternate serialization of the original Infoset. The receiving application can reconstruct the XML Infoset from the XOP package by deserialising the contents for further manipulations as shown by step 4 in Figure 1.

4.10.1 Technical implementation details and examples

It is important to note that normally in most applications, binary data need never be encoded in *base64* form. If the data is available as binary, then the application can directly copy that data into a XOP package, at the same time preparing suitable linking elements for use in the root part; when parsing a XOP package, the binary data can be made available directly to applications, or, if appropriate, the *base64* binary character representation can be computed from the binary data.

If you look at the structure of MTOM, you'll see it combines the composability of *base64* encoding with the transport efficiency of SOAP with attachments. Since the data lives within the SOAP envelope, all the WS-* specifications that Microsoft has developed with its partners will work seamlessly with MTOM. Transport efficiency with MTOM is increased with the fact that opaque data can be sent in its raw form without *base64* encoding. The opaque data is inherently dealt with and is simply streamed as binary data as one of the MIME message parts

MTOM defines a SOAP feature, i.e. an abstract piece of functionality extending the SOAP messaging framework that may be implemented to take advantage of the XOP optimizations for SOAP messages. The HTTP SOAP transmission optimization feature defines and enhances the SOAP HTTP binding by carrying the SOAP message on the wire as a XOP package.

Now we are going to take an example and show you how the optimization is achieved using the MTOM specifications. Let's assume following SOAP envelope snippet is an original XML Infoset that needs to be optimised:

```
<soap:Envelope
  xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
  xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
 <soap:Body>
  <m:data xmlns:m='http://someexample.org'>
   <m:photo>/RTRfkgnflkgnf=fndslkfnkldsnfkldsnf43d5fwfsgs4ojojsfiodjs=</m:photo>
   <m:doc>GTERkdmdfmdkf=jkszndnsjdnsjkdnskjndkjsndkjsndks</m:doc>
  </m:data>
 </soap:Body>
</soap:Envelope>
```

Listing 29 - Sample SOAP message containing binary data

If you look at the elements in the sample you can identify that there are two binary structures to work on. One is represented by the <*photo*> element and the other is the <*doc*> element. When you send this message with MTOM, the root of the XOP package will be the SOAP envelope with the photo and the document attached in the other parts of the MIME package, appropriately based on the XOP optimization procedure. So the message looks like the following:

```
MIME-Version: 1.0

Content-Type: Multipart/Related;boundary=MIME_boundary;
type="application/xop+xml";
start="<mymessage.xml@ someexample.org>";
startinfo="application/soap+xml; action="ProcessData""
Content-Description: A SOAP message with a photo and document in it

—MIME_boundary
Content-Type: application/xop+xml;
charset=UTF-8;
type="application/soap+xml; action="ProcessData""
Content-Transfer-Encoding: 8bit
Content-ID: <mymessage.xml@someexample.org>

<soap:Envelope
   xmlns:soap=”http://www.w3.org/2003/05/soap-envelope”
   xmlns:xmlmime=”http://www.w3.org/2004/11/xmlmime”>
 <soap:Body>
   <m:data xmlns:m=”http:// someexample.org/stuff”>
     <m:photo><xop:Include  xmlns:xop=”http://www.w3.org/2004/08/xop/include”
       href=”cid:http:// someexample.org/photo.jpeg” /></m:photo>
     <m:doc><xop:Include xmlns:xop=”http://www.w3.org/2004/08/xop/include”
       href=”cid:http:// someexample.org/my.worddoc” /></m:doc>
   </m:data>
 </soap:Body>
</soap:Envelope>

—MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <http:// someexample.org/photo.jpg>

// binary octets for JPEG image

—MIME_boundary
Content-Type: application/msword
Content-Transfer-Encoding: binary
Content-ID: <http://someexample.org/my.worddoc>

// binary octets for document

—MIME_boundary—
```

Listing 30 - MTOM+XOP MIME transformation of the previous example message

Binary data is referenced inside the message using the *<xop:Include>* element. Here it contains a *href* attribute which is a representation of a URI referencing the part of the package containing the data logically included by the *xop:Include* element. The *href* value must be a valid URI per the Content-ID:

URI scheme. In the message fragment shown above for the *xop:Include* element, consider this bit of code:

```
<xop:Include xmlns:xop=”http://www.w3.org/2004/08/xop/include”
   href=”cid:http://someexample.org/my.worddoc” />
```

Listing 31 - XOP include element

This will have a corresponding MIME part with a Content-ID header field with a field value as shown below.

```
Content-ID: <http://someexample.org/my.worddoc>

// binary octets for document
```

Listing 32 - MIME referenced part

This explains the simple way the binary octet data is referenced in the message stream. MTOM thus enables the referencing of binary data as it is without the requirement to do any encoding. Note also that the sample *base64* data is smaller than would be typically and the binary octets are not shown; in practice, **the optimized form is likely to be much smaller than the original**.

4.10.2 Influence on INSPIRE infrastructure

INSPIRE servers need to choose whether to accept anyway non MTOM-encoded requests, or to limit valid requests to MTOM coded ones. In the first case, sample messages like the ones that will be showed in sections 5.1 and 5.2 would be accepted anyway, else only requests similar to the one shown in Listing 34 will be accepted.
As being a server configuration option totally transparent to the single Web methods, it is not possible to read such a preference in the service WSDL, so it will be needed to specify a particular parameter in the *ExtendedCapabilities* section of the *Capabilities* document, to make this server choice available to the users.

Such a parameter could be of the following type:

```
<s:complexType name="ServerOptions">
   <s:annotation>
      <s:documentation>Data-type to specify various server options</s:documentation>
   </s:annotation>
   <s:sequence>
      <s:element name="AcceptOnlyMTOM" type="s:boolean" />
   </s:sequence>
</s:complexType>
```

Listing 33 - Possible Server options data-type

```
--MIMEBoundary000000
content-id: <0.0000@inspire.org>
content-type: application/xop+xml; charset=utf-8; type="text/xml; charset=utf-8"
content-transfer-encoding: binary

<soap:Envelope xmlns:xop="http://www.w3.org/2004/08/xop/include"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery">
 <soap:Body>
```

```
   <tns:provaMTOM>
    <tns:data><xop:Include href="cid:1.63356223930496234 3@example.org" /></tns:data>
   </tns:provaMTOM>
  </soap:Body>
</soap:Envelope>

--MIMEBoundary000000
content-id: <1.63356223930496234 3@example.org>
content-type: application/octet-stream
content-transfer-encoding: binary

//binary data
--MIMEBoundary000000--
```

Listing 34 - Sample of a valid request for a MTOM-only server

Simple requests will receive an *HTTP 415 error* response: *"Media unsupported"* if the server does not allow simple requests.

Apart from being encapsulated inside MIME messages, SOAP responses and requests are equal to the non MTOM/XOP ones, but where *base64*-encoded data is required, the element value is substituted with

```
<xop:Include href="cid:X.XXXXXXXXXX@example.org" />
```

Listing 35 - XOP include element

And *X.XXXXXXXXXX@example.org* will be the *content-id* of a MIME attachment of the same message.

5. Discovery and View services WSDL in depth

In this section we are going to inspect the single WSDLs of *Discovery* and *View* services, looking at the single operations defined, comparing their INSPIRE SOAP implementation with the OGC one and analysing how the final WSDL has been created, giving a glance at input and output parameters, data types and exceptions. Moreover, SOAP requests and responses messages are illustrated, showing how to communicate with the SOAP INSPIRE web-services.

5.1 Discovery Service

To begin with the examination of the *Discovery Service*, we are going to illustrate the single methods that compose this INSPIRE service, going then in depth into each one of those, looking at the mapping between OGC parameters and INSPIRE ones and consequently how the WSDL fragments for the single operations are generated.

5.1.1 INSPIRE Discovery Service operations

The *Network Services Drafting Team* made a proposal for the technical content of the INSPIRE Implementing Rules for Discovery Service; such proposal is available and described in "Draft Technical Guidance Document for INSPIRE Discovery Services" [INSPIRE TechG Disc/1.0].

Hereafter we report the mapping between the operations proposed by [OGC CSW] and the INSPIRE *Discovery Service* operations (in the cardinality column, 'M' stands for Mandatory, while 'O' stands for Optional):

INSPIRE Discovery Services functions	*INSPIRE Cardinality*	*OGC CSW ISO AP operations*
INSPIRE.GetDiscoveryServiceMetadata	M	OGC_Service.GetCapabilities
INSPIRE.DiscoverMetadata	M	CSW Discovery.GetRecords
INSPIRE.GetMetadata	M	CSW Discovery.GetRecords
INSPIRE.PublishMetadata	M, O	CSWT Manager.Transaction
INSPIRE.CollectMetadata	M, O	CSWT Manager.Harvest

Table 2 - INSPIRE Discovery service and OGC ISO AP operations mapping

An INSPIRE *Discovery Service* must support at least the operations indicated as "mandatory" in Table 2. Thus, an INSPIRE *Discovery Service* is always a "read-only" OGC Catalogues Service as defined by [OGC CSW], but not a fully "transactional" OGC Catalogues Service: only one of both transactional operations (*Manager.Transaction* or *Manager.Harvest*) is mandatory.

Accordingly with the INSPIRE SOAP Framework [EUR 23635 - 2008] all operations will support the embedding of requests and responses in SOAP 1.1 messages with Document/literal wrapped style.

5.1.1.1 GetDiscoveryServiceMetadata

The mandatory *GetDiscoveryServiceMetadata* operation allows any client to retrieve metadata about the capabilities provided by the INSPIRE server. The normal response to the *GetDiscoveryServiceMetadata* operation is a service metadata document that is returned to the requesting client. This service metadata document primarily contains metadata about the specific server abilities (such as about the specific data and formats available from that server) and also makes a server partially self-describing, supporting late binding of clients. Some types are service instance specific and therefore cannot be statically specified in the WSDL level, therefore such operation allows the requesting client to have a full understanding of the contacted server instance and of its services.

The WSDL *portType* component of the *OGC_Service* interface is shown in the following Listing; this is a fragment of the complete WSDL 2.0 definition for the CSW Catalogue made by OWS [07-045].

```
<wsdl:portType name="csw">
  <wsdl:operation name="csw.getCapabilities">
    <wsdl:input message="csw-req:GetCapabilitiesRequest"/>
    <wsdl:output message="csw-resp:GetCapabilitiesResponse"/>
    <wsdl:fault name="ServiceExceptionReport" message="csw-resp:ServiceExceptionReport"/>
  </wsdl:operation>
</wsdl:portType>
```

```
<wsdl:operation name="csw.getCapabilities">
  <soap:operation soapAction="http://www.opengis.net/cat/csw/2.0.2/requests#GetCapabilities"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceExceptionReport">
    <soap:fault use="literal" name="ServiceExceptionReport"/>
  </wsdl:fault>
</wsdl:operation>
```

Listing 36 - WSDL getCapabilities interface

5.1.1.1.1 OGC Request

The OGC *GetCapabilities* operation request is defined in Subclause 7.2 of the *OGC Web Services Common Specification 1.0* [05-008c1].
The OGC *GetCapabilities* operation foresaw several optional and compulsory input parameters, but none of these are anymore necessary in the INSPIRE implementation, as recommended in [INSPIRE TechG Disc/1.0].

This is the OGC method WSDL request message structure:

```
<wsdl:message name="GetCapabilitiesRequest">
  <wsdl:part name="Body" element="csw:GetCapabilities"/>
</wsdl:message>
```

Listing 37 - OGC GetCapabilities WSDL request message structure

We do not report here after the *cws:GetCapabilities* type, because it is not useful for the INSPIRE input service definition, as much as the INSPIRE method does not need any input parameter as previously explained.

5.1.1.1.2 OGC Response

The response to a *GetCapabilities* request should be an XML document containing service metadata about the server, as defined in [05-008c1] and [07-006].

If we explore the response of the service, several OGC output parameters are relevant for the INSPIRE *GetDiscoveryServiceMetadataResponse* too:

INSPIRE.GetDiscoveryServiceMetadata Output	*CSW ISO AP compulsory output*
ServiceType element	ServiceIdentification: ServiceType, ServiceTypeVersion, Title, Abstract, Keywords, Fees, AccessConstraints
ServiceProvider element	ServiceProvider: ProviderName, Providersite, ServiceContact
SupportedFunctions element	OperationsMetadata: Operation, Parameter, Constraint, ExtendedCapabilities
QueryLanguage element	Filter_Capabilities: And, Or, Not, PropertyIsEqualTo, PropertyIsNotEqualTo, PropertyIsLessThan, PropertyIsGreaterThan, PropertyIsLessThanOrEqualTo, PropertyIsGreaterThanOrEqualTo, BBOX.
SupportedMetadataLanguages element	OperationsMetadata section: ExtendedCapabilities

Table 3 - Mapping between INSPIRE GetDiscoveryServiceMetadata interesting output parameters and OGC GetCapability ones

This is the OGC method WSDL response message structure:

```
<wsdl:message name="GetCapabilitiesResponse">
  <wsdl:part element="csw:Capabilities" name="Body"/>
</wsdl:message>
```

Listing 38 - OGC GetCapabilities WSDL response message structure

And here after we report the data and data types used in the response:

```
<xsd:element name="Capabilities" id="Capabilities" type="csw:CapabilitiesType"/>
 <xsd:complexType name="CapabilitiesType" id="CapabilitiesType">
  <xsd:annotation>
   <xsd:documentation>This type extends ows:CapabilitiesBaseType defined in OGC-05-008
   to include information about supported OGC filter components. A profile may extend this type to describe
   additional capabilities.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:extension base="ows:CapabilitiesBaseType">
    <xsd:sequence>
     <xsd:element ref="ogc:Filter_Capabilities"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

```
<complexType name="CapabilitiesBaseType">
  <annotation>
   <documentation>XML encoded GetCapabilities operation response. This document provides clients with service
   metadata about a specific service instance, usually including metadata about the tightly-coupled data served.
   If the server does not implement the updateSequence parameter, the server shall always return the complete
   Capabilities document, without the updateSequence parameter. When the server implements the
   updateSequence parameter and the GetCapabilities operation request included the updateSequence
   parameter with the current value, the server shall return this element with only the "version" and
   "updateSequence" attributes. Otherwise, all optional elements shall be included or not depending on the
   actual value of the Contents parameter in the GetCapabilities operation request. This base type shall be
   extended by each specific OWS to include the additional contents needed.
   </documentation>
  </annotation>
  <sequence>
   <element ref="ows:ServiceIdentification" minOccurs="0"/>
   <element ref="ows:ServiceProvider" minOccurs="0"/>
   <element ref="ows:OperationsMetadata" minOccurs="0"/>
  </sequence>
  <attribute name="version" type="ows:VersionType" use="required"/>
  <attribute name="updateSequence" type="ows:UpdateSequenceType" use="optional">
   <annotation>
    <documentation>Service metadata document version, having values that are "increased" whenever any
    change is made in service metadata document. Values are selected by each server, and are always opaque
    to clients. When not supported by server, server shall not return this attribute. </documentation>
   </annotation>
  </attribute>
</complexType>
```

Listing 39 - OGC GetCapabilities response data and data types

We think it could be useful to report in this section even the *Filter_Capabilities* element composition, because it will be used in the INSPIRE *GetDiscoveryServiceMetadata* response:

```
<xsd:element name="Filter_Capabilities">
  <xsd:complexType>
```

```
    <xsd:sequence>
      <xsd:element name="Spatial_Capabilities" type="ogc:Spatial_CapabilitiesType"/>
      <xsd:element name="Scalar_Capabilities" type="ogc:Scalar_CapabilitiesType"/>
      <xsd:element name="Id_Capabilities" type="ogc:Id_CapabilitiesType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Listing 40 - OGC Filter_Capabilities element structure

As reported in [INSPIRE TechG Disc/1.0], the following XSD code defines the XSD Types that are needed to provide additional information on multilingual aspects. This information has to be provided in a capabilities documents that is returned by an INSPIRE *Discovery Service*.

The XML Elements that comply with the following shall be applied in the <*ExtendedCapabilities*> section of the capabilities document.

```
<xs:complexType name="InspireCapabilitiesType">
  <xs:annotation>
    <xs:documentation>Additional capabilities for INSPIRE</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Languages" type="LanguagesType" minOccurs="0"/>
    <xs:element name="TranslatedCapabilities" type="TranslatedCapabilitiesType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="MultiligualCapabilities" type="InspireCapabilitiesType"></xs:element>
<xs:complexType name="LanguagesType">
  <xs:annotation>
    <xs:documentation>
      List of languages defined by a 3-letter code as described in ISO 639-2 that are supported by this service
      instance.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Language" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="TranslatedCapabilitiesType">
  <xs:annotation>
    <xs:documentation>
      List of URLs that give access to translation of capabilties document at hand.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Document" type="DocumentType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="DocumentType">
```

```
    <xs:annotation>
      <xs:documentation>
        Connect point URL to translanted capabilities document. The language attribute shall be defined by a 3-
        letter code as described in ISO 639-2.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="ows:OnlineResourceType">
        <xs:attribute name="language" use="required"></xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

Listing 41 – MultilingualCapabilities data-type

5.1.1.1.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data style and encoding to define the WSDL morphology and so, for this operation, we obtain the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows=" http://www.opengis.net/ows/1.1"
  xmlns:ogc=" http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >

  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
      <s:element name="getCapabilities">
        <s:complexType/>
      </s:element>
      <s:element name="GetDiscoveryServiceMetadataResponse">
        <s:complexType>
          <s:sequence>
            <s:element ref="ows:ServiceIdentification" />
            <s:element ref="ows:ServiceProvider" />
            <s:element ref="ogc:Filter_Capabilities" />
            <s:element ref="ows:OperationsMetadata" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>

    <s:complexType name="ServerOptions">
```

```
        <s:annotation>
          <s:documentation>Data-type to specify various server options</s:documentation>
        </s:annotation>
        <s:sequence>
          <s:element name="AcceptOnlyMTOM" type="s:boolean" />
        </s:sequence>
      </s:complexType>
    </wsdl:types>

  <wsdl:message name="GetDiscoveryServiceMetadataSoapIn">
    <wsdl:part name="parameters" element="tns:getCapabilities" />
  </wsdl:message>
  <wsdl:message name="GetDiscoveryServiceMetadataSoapOut">
    <wsdl:part name="parameters" element="tns:GetDiscoveryServiceMetadataResponse" />
  </wsdl:message>

  <wsdl:portType name="DiscoveryServiceSoap">
    <wsdl:operation name="GetDiscoveryServiceMetadata">
      <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Request to a Service to perform the
        GetCapabilities operation. This operation allows a client to retrieve a Capabilities XML document providing
        metadata for the specific INSPIRE server.</wsdl:documentation>
      <wsdl:input message="tns:GetDiscoveryServiceMetadataSoapIn" />
      <wsdl:output message="tns:GetDiscoveryServiceMetadataSoapOut" />
      <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
    </wsdl:operation>
  </wsdl:portType>
```

```
<wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
    <wsdl:documentation>
      <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
        xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
    </wsdl:documentation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetDiscoveryServiceMetadata">
      <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/GetDiscoveryServiceMetadata"
       style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
        <soap:body use="literal" name="ServiceExceptionReport" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

</wsdl:definitions>
```

Listing 42 - GetDiscoveryServiceMetadata WSDL parts

5.1.1.1.4 SOAP Request Message

Here after we report a sample SOAP/1.1 request message for the *GetDiscoveryServiceMetadata* operation:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <getCapabilities xmlns="http://inspire.jrc.ec.europa.eu/Discovery"/>
  </soap:Body>
</soap:Envelope>
```

Listing 43 - Sample GetDiscoveryServiceMetadata SOAP request message

As we can see from this piece of code, no parameters are required, so the SOAP message is as straightforward as possible.

5.1.1.1.5 SOAP Response message

Here after we reproduced a sample SOAP/1.1 response message for the *GetDiscoveryServiceMetadata* operation:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ows="http://www.opengis.net/ows" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:csw="http://www.opengis.net/cat/csw">
   <soap:Body>
     <GetDiscoveryServiceMetadataResponse xmlns=”http://inspire.jrc.ec.europa.eu/Discovery”
        updateSequence="0" xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ns0="http://www.opengis.net/cat/csw" xmlns:ns1="http://www.opengis.net/cat/csw">
      <GetDiscoveryServiceMetadataResult>
       <ows:ServiceIdentification>
         <ows:ServiceType>CSW</ows:ServiceType>
         <ows:ServiceTypeVersion>2.0.0</ows:ServiceTypeVersion>
         <ows:Title>Company CSW</ows:Title>
         <ows:Abstract>
           A catalogue service that conforms to the HTTP protocol binding of the OpenGIS Catalogue Service
          specification version 2.0.0.
         </ows:Abstract>
         <ows:Keywords>
           <ows:Keyword>CSW</ows:Keyword>
           <ows:Keyword>Company Name</ows:Keyword>
           <ows:Keyword>geospatial</ows:Keyword>
           <ows:Keyword>catalogue</ows:Keyword>
         </ows:Keywords>
         <ows:Fees>NONE</ows:Fees>
         <ows:AccessConstraints>NONE</ows:AccessConstraints>
       </ows:ServiceIdentification>
       <ows:ServiceProvider>
         <ows:ProviderName>Company Name</ows:ProviderName>
         <ows:ProviderSite ans1:href="http://www.oracle.com" xmlns:ans1="http://www.w3.org/1999/xlink"/>
         <ows:ServiceContact>
           <ows:IndividualName> Contact Person Name</ows:IndividualName>
           <ows:PositionName>Staff</ows:PositionName>
           <ows:ContactInfo>
             <ows:Phone>
               <ows:Voice>999-999-9999</ows:Voice>
               <ows:Facsimile>999-999-9999</ows:Facsimile>
             </ows:Phone>
           <ows:Address>
             <ows:DeliveryPoint>1 Street Name</ows:DeliveryPoint>
             <ows:City>CityName</ows:City>
             <ows:AdministrativeArea>StateName</ows:AdministrativeArea>
             <ows:PostalCode>09999</ows:PostalCode>
             <ows:Country>USA</ows:Country>
```

```
            <ows:ElectronicMailAddress>contact.person@company.com</ows:ElectronicMailAddress>
          </ows:Address>
          <ows:OnlineResource ans1:href="mailto:contact.person@company.com"
           xmlns:ans1="http://www.w3.org/1999/xlink"/>
        </ows:ContactInfo>
      </ows:ServiceContact>
    </ows:ServiceProvider>
    <ogc:Filter_Capabilities>
      <ogc:Spatial_Capabilities>
        <ogc:Spatial_Operators>
          <ogc:BBOX/>
          <ogc:Equals/>
          <ogc:Disjoint/>
          <ogc:Intersect/>
          <ogc:Touches/>
          <ogc:Crosses/>
          <ogc:Within/>
          <ogc:Contains/>
          <ogc:Overlaps/>
          <ogc:Beyond/>
          <ogc:DWithin/>
        </ogc:Spatial_Operators>
      </ogc:Spatial_Capabilities>
      <ogc:Scalar_Capabilities>
        <ogc:Logical_Operators/>
        <ogc:Comparison_Operators>
          <ogc:Simple_Comparisons/>
          <ogc:Like/>
          <ogc:Between/>
          <ogc:NullCheck/>
        </ogc:Comparison_Operators>
        <ogc:Arithmetic_Operators>
          <ogc:Simple_Arithmetic/>
        </ogc:Arithmetic_Operators>
      </ogc:Scalar_Capabilities>
    </ogc:Filter_Capabilities>
    <ows:OperationsMetadata>
      <ows:Operation name="GetCapabilities">
        <ows:DCP>
          <ows:HTTP>
            <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
             root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
          </ows:HTTP>
        </ows:DCP>
      </ows:Operation>
      <ows:Operation name="DescribeRecord">
        <ows:DCP>
          <ows:HTTP>
```

```
            <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
              root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
          </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="typeName">
          <ows:Value>ns0:SampleRecord</ows:Value>
          <ows:Value>ns1:Record</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputFormat">
          <ows:Value>text/xml</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="schemaLanguage">
          <ows:Value>XMLSCHEMA</ows:Value>
        </ows:Parameter>
      </ows:Operation>
      <ows:Operation name="GetRecords">
        <ows:DCP>
          <ows:HTTP>
            <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
              root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
          </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="TypeName">
          <ows:Value>ns0:SampleRecord</ows:Value>
          <ows:Value>ns1:Record</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputFormat">
          <ows:Value>text/xml</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputSchema">
          <ows:Value>OGCCORE</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="resultType">
          <ows:Value>hits</ows:Value>
          <ows:Value>results</ows:Value>
          <ows:Value>validate</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="ElementSetName">
          <ows:Value>brief</ows:Value>
          <ows:Value>summary</ows:Value>
          <ows:Value>full</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="CONSTRAINTLANGUAGE">
          <ows:Value>Filter</ows:Value>
        </ows:Parameter>
      </ows:Operation>
      <ows:Operation name="GetRecordById">
        <ows:DCP>
```

```
        <ows:HTTP>
          <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
            root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="ElementSetName">
        <ows:Value>brief</ows:Value>
        <ows:Value>summary</ows:Value>
        <ows:Value>full</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetDomain">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
            root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="ParameterName">
        <ows:Value>GetRecords.resultType</ows:Value>
        <ows:Value>GetRecords.outputFormat</ows:Value>
        <ows:Value>GetRecords.outputRecType</ows:Value>
        <ows:Value>GetRecords.typeNames</ows:Value>
        <ows:Value>GetRecords.ElementSetName</ows:Value>
        <ows:Value>GetRecords.ElementName</ows:Value>
        <ows:Value>GetRecords.CONSTRAINTLANGUAGE</ows:Value>
        <ows:Value>GetRecordById.ElementSetName</ows:Value>
        <ows:Value>DescribeRecord.typeName</ows:Value>
        <ows:Value>DescribeRecord.schemaLanguage</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="Transaction">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post ans1:href="http://localhost:8888/SpatialWS-SpatialWS-context-
            root/SpatialWSSoapHttpPort" xmlns:ans1="http://www.w3.org/1999/xlink"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Parameter name="service">
      <ows:Value>CSW</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="version">
      <ows:Value>2.0.0</ows:Value>
    </ows:Parameter>
    <ows:ExtendedCapabilities>
      <MultiligualCapabilities xmlns:xlink="http://www.w3.org/1999/xlink" >
        <Languages>
```

```
            <Language>GER</Language>
            <Language>DUT</Language>
          </Languages>
          <TranslatedCapabilities>
            <Document xlink:href="http://www.somehost.com/capabilities_german.xml" language="GER"/>
            <Document xlink:href="http://www.somehost.com/capabilities_dutch.xml" language="DUT"/>
          </TranslatedCapabilities>
        </MultiligualCapabilities>
      </ows:ExtendedCapabilities>
    </ows:OperationsMetadata>
  </GetDiscoveryServiceMetadataResult>
 </GetDiscoveryServiceMetadataResponse>
</soap:Body>
</soap:Envelope>
```

Listing 44 - Sample GetDiscoveryServiceMetadata SOAP response message

As we can see from this piece of code, the method returns a *Capabilities* object, containing the INSPIRE output parameters, completed with the Multilingual capabilities of the service (<*MultiligualCapabilities*> element).

5.1.1.2 DiscoverMetadata

The *DiscoverMetadata* operation allows to request for all or a predefined set of metadata (record) elements of spatial resources, based on a query statement. These will be retrieved from the target *Discovery Service* data-store.

The WSDL *portType* component of the *csw:getRecords* interface is shown in following Listing; this is a fragment of the complete WSDL 2.0 definition for the CSW Catalogue ([CSW XML-INT] and [CSW PUBL]).

```
<wsdl:operation name="csw.getRecords">
  <wsdl:input message="csw-req:GetRecordsRequest"/>
  <wsdl:output message="csw-resp:GetRecordsResponse"/>
  <wsdl:fault name="ServiceExceptionReport" message="csw-resp:ServiceExceptionReport"/>
</wsdl:operation>
```

```
<wsdl:operation name="csw.getRecords">
  <soap:operation soapAction="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceExceptionReport">
    <soap:fault use="literal" name="ServiceExceptionReport"/>
  </wsdl:fault>
</wsdl:operation>
```

Listing 45 - WSDL CSW_getRecords interface

5.1.1.2.1 OGC Request

The following table specifies the input parameters taken from the corresponding OGC operation (*GetRecords*) request, that have been considered useful in INSPIRE domain.

INSPIRE.DiscoverMetadata Input	*CSW ISO AP compulsory input*
QueryLanguage	ConstraintLanguage[1]
Query	Constraint
SortSpec	SortBy
SortOrder *	SortOrder[2]
SearchType (hits, results)	ResultType (hits, results, validate)

Table 4 - Mapping between INSPIRE DiscoverMetadata input parameters and OGC GetRecords iportant ones

The following table aims to explain in a more detailed way the main OGC input parameters for this method; the column “ISO Metadata Profile” shows syntax and/or semantics restrictions or variations in comparison to those of the base specification:

[1] The Constraint language is defined by the element name contained in *Constraint* element.
[2] The *SortOrder* parameter is a sub-part of the *SortBy* one.

Name	*Data type and value*	*Multiplicity and use*	*ISO Metadata Profile*
CONSTRAINTLANGUAGE	**CodeList** One of “CQL_TEXT” or “FILTER”	Must be specified with *QUERYCONSTRAINT* parameter	Must be specified with QUERYCONSTRAINT parameter
CONSTRAINT	**String** The predicate expression specified in the language indicated by the CONSTRAINTLANGUAGE parameter	*Optional*	*Optional* Default action is to execute an unconstrained query
SORTBY	**List of Character String, comma separated** Ordered list of names of metadata elements to use for sorting the response. Format of each list item is *metadata_element_name:A* indicating an ascending sort or *metadata_element_name:D* indicating descending sort. *metadata_element_name*: use only the plain name (not case sensitive) without any prefixes etc, because these are uniquely defined. Example: Denominator instead of SpatialResolution.Denominator	*Optional*	*Optional* Default action is to present the records in the order in which they are retrieved
RESULTTYPE	**CodeList** One of “hits”, “results” or “validate”	*Optional*	*Optional* Default value is “hits”. Indicate whether the Catalogue returns the full result set (if ELEMENTSETNAME or ELEMENTNAME are missing) or just the number of hits the query found. If the value is “hits”, ELEMENTSETNAME or ELEMENTNAME are ignored

Table 5 - Compulsory Parameters in OGC GetCapabilities operation request

Here after we report the OGC method WSDL request message structure and the correlated data and data-types used:

```
<wsdl:message name="GetRecordsRequest">
  <wsdl:part name="Body" element="csw:GetRecords"/>
</wsdl:message>
```

```
<xsd:element name="GetRecords" type="csw:GetRecordsType" id="GetRecords"/>
 <xsd:complexType name="GetRecordsType" id="GetRecordsType">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
   The principal means of searching the catalogue. The matching catalogue entries may be included with the response. The client may assign a requestId (absolute URI). A distributed search is performed if the DistributedSearch element is present and the catalogue is a member of a federation. Profiles may allow alternative query expressions.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:extension base="csw:RequestBaseType">
    <xsd:sequence>
     <xsd:element name="DistributedSearch" type="csw:DistributedSearchType" minOccurs="0"/>
     <xsd:element name="ResponseHandler" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
     <xsd:choice>
      <xsd:element ref="csw:AbstractQuery"/>
      <xsd:any processContents="strict" namespace="##other"/>
     </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="requestId" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="resultType" type="csw:ResultType" use="optional" default="hits"/>
    <xsd:attributeGroup ref="csw:BasicRetrievalOptions"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

Listing 46 - OGC GetRecords WSDL. Request message structure

5.1.1.2.2 OGC Response

This operation must respond with an XML document including the method results. If the *resultType* parameter is set to "results", the catalogue service must include any matching records within the <*SearchResults*> element.

In the following table we can see the INSPIRE relevant response parameters:

INSPIRE.GetDiscoverMetadata Output	*CSW ISO AP compulsory output*
RequestValidity	SearchStatus
ResultCount	SearchResults (numberOfRecordsMatched)
ResultMetadataIDList	SearchResults (resultSetID)

Table 6 - Mapping between INSPIRE GetDiscoverMetadata output parameters and OGC GetRecords ones

This is the OGC method WSDL response message structure, together with the data and data types used:

```
<wsdl:message name="GetRecordsResponse">
  <wsdl:part element="csw:GetRecordsResponse" name="Body"/>
</wsdl:message>
```

```
<xsd:element name="GetRecordsResponse" type="csw:GetRecordsResponseType"
    id="GetRecordsResponse"/>
<xsd:complexType name="GetRecordsResponseType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The response message for a GetRecords request. Some or all of the matching records may be included as
      children of the SearchResults element. The RequestId is only included if the client specified it.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="RequestId" type="xsd:anyURI" minOccurs="0"/>
    <xsd:element name="SearchStatus" type="csw:RequestStatusType"/>
    <xsd:element name="SearchResults" type="csw:SearchResultsType"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Listing 47 - OGC GetRecords WSDL. Response message structure

The <*GetRecordsResponse*> element is a container for the response of the *GetRecords* operation. Three levels of detail may be contained in the response document.
The <*RequestId*> element may be used to correlate the response to a *GetRecords* request for which a value was defined for the *requestId* attribute.
The <*SearchStatus*> element shall be present and indicates the status of the response: it consists of a timestamp attribute indicating when the result set was created.
The <SearchResults> element is a generic XML container for the actual response to a *GetRecords* request: its content is the set of records returned by the *GetRecords* operation.

5.1.1.2.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data encoding, obtaining the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows=" http://www.opengis.net/ows/1.1"
 xmlns:ogc=" http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
 xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >

 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
   <s:element name="DiscoverMetadataRequest">
    <s:complexType>
     <s:sequence>
      <s:element ref="csw:Constraint" />
      <s:element name="SortBy" type="ogc:SortByType" />
    </s:sequence>
    <s:attribute name="resultType" type="csw:ResultType" use="optional" default="hits" />
    </s:complexType>
   </s:element>
   <s:element name="DiscoverMetadataResponse">
    <s:complexType>
     <s:sequence>
      <s:element name="SearchStatus" type="csw:RequestStatusType" />
      <s:element name="SearchResult" type="csw:SearchResultType" />
     </s:sequence>
    </s:complexType>
   </s:element>
  </s:schema>
 </wsdl:types>

 <wsdl:message name="DiscoverMetadataSoapIn">
  <wsdl:part name="parameters" element="tns: DiscoverMetadataRequest " />
 </wsdl:message>
 <wsdl:message name="DiscoverMetadataSoapOut">
  <wsdl:part name="parameters" element="tns:DiscoverMetadataResponse" />
 </wsdl:message>

 <wsdl:portType name="DiscoveryServiceSoap">
  <wsdl:operation name="DiscoverMetadata">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">DiscoverMetadata service
   description.</wsdl:documentation>
```

```
    <wsdl:input message="tns:DiscoverMetadataSoapIn" />
    <wsdl:output message="tns:DiscoverMetadataSoapOut" />
    <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
   </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
   <wsdl:documentation>
    <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1" xmlns:wsi="http://ws-
    i.org/schemas/conformanceClaim/" />
   </wsdl:documentation>
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="DiscoverMetadata">
    <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/DiscoverMetadata" style="document"
    />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
    <wsdl:fault name="ServiceExceptionReport">
       <soap:body use="literal" name="ServiceExceptionReport" />
    </wsdl:fault>
   </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

Listing 11 – DiscoverMetadata operation WSDL parts

5.1.1.2.4 SOAP Request message

Here after we produced a sample SOAP/1.1 request message for the *DiscoverMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc=" http://www.opengis.net/ogc"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
 <soap:Body>
  <DiscoverMetadataRequest xmlns="http://inspire.jrc.ec.europa.eu/Discovery" resultType="results"/>
      <csw:Constraint version="2.0.2" >
         <ogc:Filter>
           <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>/csw:Record/dc:contributor</ogc:PropertyName>
            <ogc:Literal>Raja</ogc:Literal>
           </ogc:PropertyIsEqualTo>
         </ogc:Filter>
      </csw:Constraint>
       <SortBy>Denominator:A</SortBy>
  </DiscoverMetadata>
 </soap:Body>
</soap:Envelope>
```

Listing 48 - Sample DiscoverMetadata SOAP request message

5.1.1.2.5 SOAP Response message

Here after we produced a sample SOAP/1.1 response message for the *DiscoverMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
 <soap:Body>
  <DiscoverMetadataResponse xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
   <csw:SearchStatus status="complete"/>
   <csw:SearchResults recordSchema="http://www.opengis.net/cat/csw" numberOfRecordsMatched="1"
     numberOfRecordsReturned="1" nextRecord="0" expires="2007-02-09T16:32:35.29Z">
      <csw:Record xmlns:dc="http://www.purl.org/dc/elements/1.1/" xmlns:ows="http://www.opengis.net/ows"
     xmlns:dct="http://www.purl.org/dc/terms/">
        <dc:contributor xmlns:dc="http://www.purl.org/dc/elements/1.1/"
       scheme="http://www.altova.com">Raja</dc:contributor>
        <dc:identifier xmlns:dc="http://www.purl.org/dc/elements/1.1/">REC-1</dc:identifier>
      </csw:Record>
   </csw:SearchResults>
  </DiscoverMetadataResponse>
 </soap:Body>
</soap:Envelope>
```

Listing 49 - Sample DiscoverMetadata SOAP response message

5.1.1.3 GetMetadata

The mandatory *GetMetadata* operation allows to retrieve metadata for specific resources from a result set based on the resource unique Identification (*ID*).

The WSDL *portType* component of the *getRecordById* interface is shown in following Listing; this is a fragment of the complete WSDL 2.0 definition for the CSW Catalogue ([CSW XML-INT] and [CSW PUBL]).

```
<wsdl:operation name="csw.getRecordById">
  <wsdl:input message="csw-req:GetRecordByIdRequest"/>
  <wsdl:output message="csw-resp:GetRecordByIdResponse"/>
  <wsdl:fault name="ServiceExceptionReport" message="csw-resp:ServiceExceptionReport"/>
</wsdl:operation>
```

```
<wsdl:operation name="csw.getRecordById">
  <soap:operation soapAction="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecordsById"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceExceptionReport">
    <soap:fault use="literal" name="ServiceExceptionReport"/>
  </wsdl:fault>
</wsdl:operation>
```

Listing 50 - WSDL CSW_GetRecordById interface

5.1.1.3.1 OGC Request

The *GetRecordById* request retrieves a representation of one or more specific catalogue records using their identifier(s).
The following table specifies the input parameters taken from the OGC operation request (right column), that have been considered useful for INSPIRE domain [INSPIRE TechG Disc/1.0].

INSPIRE.GetMetadata Input	*CSW ISO AP compulsory input*
MetadataElementList	ElementSetName
MetadataIdList	Id

Table 7 - Mapping between INSPIRE GetMetadata input parameters and OGC GetRecordById ones

The following table aims to explain in a more detailed way the two used OGC input parameters:

Name	*Data type and value*	*Multiplicity and use*	*ISO Metadata Profile*
ELEMENTSETNAME	**CodeList** allowed values: "brief", "summary" or "full"	Zero or one (**Optional**) Default value is "summary"	Zero or one (**Optional**) Default value is "summary"
ID	**Comma separated list** of *anyURI*	**Mandatory**	**Mandatory** identifier

Table 8 - INSPIRE interested GetRecordById input parameters description

Here after we report the *GetRecordById* WSDL request message structure and the correlated data and data-types used:

```
<wsdl:message name="GetRecordByIdRequest">
  <wsdl:part name="Body" element="csw:GetRecordById"/>
</wsdl:message>
```

```
<xsd:element name="GetRecordById" type="csw:GetRecordByIdType" id="GetRecordById"/>
 <xsd:complexType name="GetRecordByIdType" id="GetRecordByIdType">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
    Convenience operation to retrieve default record representations by identifier.
    Id - object identifier (a URI) that provides a reference to a catalogue item (or a result set if the catalogue
      supports persistent result sets).
    ElementSetName - one of "brief, "summary", or "full"
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:extension base="csw:RequestBaseType">
    <xsd:sequence>
     <xsd:element name="Id" type="xsd:anyURI" maxOccurs="unbounded"/>
     <xsd:element ref="csw:ElementSetName" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="outputFormat" type="xsd:string" use="optional" default="application/xml"/>
    <xsd:attribute name="outputSchema" type="xsd:anyURI" use="optional"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

Listing 51 - OGC GetRecordById WSDL. Request message structure

5.1.1.3.2 OGC Response

The response to a *GetRecordById* operation is simply the list of requested records.
In the following table we see the mapping between INSPIRE and OGC relevant response parameters:

INSPIRE.GetMetadata Input	*CSW ISO AP compulsory input*
GetMetadataResponse	GetRecordByIdResponse

Table 9 - Mapping between INSPIRE GetMetadata output parameter and OGC GetRecordById one

This is the OGC method WSDL response message structure, together with the data and data types used:

```
<xsd:element name="GetRecordByIdResponse"
 type="csw:GetRecordByIdResponseType" id="GetRecordByIdResponse"/>
 <xsd:complexType name="GetRecordByIdResponseType" id="GetRecordByIdResponseType">
   <xsd:annotation>
     <xsd:documentation xml:lang="en">Returns a representation of the matching entry. If there is no matching
       record, the response message must be empty.</xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
     <xsd:choice>
       <xsd:element ref="csw:AbstractRecord" minOccurs="0" maxOccurs="unbounded"/>
       <xsd:any processContents="strict" namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
     </xsd:choice>
   </xsd:sequence>
 </xsd:complexType>
```

Listing 52 - OGC GetRecordById WSDL. Response message structure

5.1.1.3.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data encoding, obtaining the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows=" http://www.opengis.net/ows/1.1"
 xmlns:ogc=" http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
 xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
<wsdl:types>
 <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
  <s:element name="GetMetadataRequest">
   <s:complexType>
    <s:sequence>
     <s:element name="elementSetName" type="csw:elementSetName" />
     <s:element name="id" type="s:anyURI" />
    </s:sequence>
   </s:complexType>
  </s:element>
  <s:element name="GetMetadataResponse" type="csw:GetRecordByIdResponseType" / >
 </s:schema>
</wsdl:types>

<wsdl:message name="GetMetadataSoapIn">
 <wsdl:part name="parameters" element="tns: GetMetadataRequest" />
</wsdl:message>
<wsdl:message name="GetMetadataSoapOut">
 <wsdl:part name="parameters" element="tns:GetMetadataResponse" />
</wsdl:message>

<wsdl:portType name="DiscoveryServiceSoap">
 <wsdl:operation name="GetMetadata">
  <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"> The GetMetadata operation allows to
  retrieve metadata for specific resources from a result set based on the resource unique Identification
  (ID).</wsdl:documentation>
  <wsdl:input message="tns:GetMetadataSoapIn" />
  <wsdl:output message="tns:GetMetadataSoapOut" />
  <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
 </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
 <wsdl:documentation>
```

```
    <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1" xmlns:wsi="http://ws-
    i.org/schemas/conformanceClaim/" />
   </wsdl:documentation>
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="GetMetadata">
    <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/GetMetadata" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
    <wsdl:fault name="ServiceExceptionReport">
      <soap:body use="literal" name="ServiceExceptionReport" />
    </wsdl:fault>
   </wsdl:operation>
  </wsdl:binding>

</wsdl:definitions>
```

Listing 53 - GetMetadata operation WSDL parts

5.1.1.3.4 SOAP Request Message

Here after we produced a sample SOAP/1.1 request message for the *GetMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:csw=”http://www.opengis.net/cat/csw/2.0.2”>
 <soap:Body>
  <GetMetadataRequest xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
   <csw:Id>REC-1</csw:Id>
   <csw:ElementSetName>brief</csw:ElementSetName>
  </GetMetadataRequest>
 </soap:Body>
</soap:Envelope>
```

Listing 54 - Sample GetMetadata SOAP request message

5.1.1.3.5 SOAP Response Message

Here after we produced a sample SOAP/1.1 response message for the *GetMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2">
 <soap:Body>
    <GetMetadataResponse xmlns:dc="http://www.purl.org/dc/elements/1.1/"
    xmlns:dct="http://www.purl.org/dc/terms/" xsi:schemaLocation="http://www.opengis.net/cat/csw
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <csw:BriefRecord xmlns:dc="http://www.purl.org/dc/elements/1.1/"
        xmlns:ows="http://www.opengis.net/ows" xmlns:dct="http://www.purl.org/dc/terms/">
        <dc:identifier xmlns:dc="http://www.purl.org/dc/elements/1.1/">REC-1</dc:identifier>
      </csw:BriefRecord>
    </GetMetadataResponse>
 </soap:Body>
</soap:Envelope>
```

Listing 55 - Sample GetMetadata SOAP response message

5.1.1.4 PublishMetadata

The *PublishMetadata* operation allows creating, deleting, or updating (set) metadata (record) elements of spatial resources in the *Discovery Service* data-store.

The WSDL *portType* component of the *transaction* interface is shown in following Listing; this is a fragment of the complete WSDL 2.0 definition for the CSW Catalogue ([CSW XML-INT] and [CSW DISC]).

```
<wsdl:operation name="csw.transaction">
  <wsdl:input message="csw-req:TransactionRequest"/>
  <wsdl:output message="csw-resp:TransactionResponse"/>
  <wsdl:fault name="ServiceExceptionReport" message="csw-resp:ServiceExceptionReport"/>
</wsdl:operation>
```

```
<wsdl:operation name="csw.transaction">
  <soap:operation soapAction="http://www.opengis.net/cat/csw/2.0.2/requests#Transaction"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceExceptionReport">
    <soap:fault use="literal" name="ServiceExceptionReport"/>
  </wsdl:fault>
</wsdl:operation>
```

Listing 56 - WSDL CSW_Transaction interface

5.1.1.4.1 OGC Request

The XML encoding of a valid response is specified in the HTTP binding of [07-006].
Here after we report the mapping between the CSW ISO AP input parameters and the INSPIRE useful ones as defined in [INSPIRE TechG Disc/1.0].

INSPIRE.PublishMetadata Input	*CSW ISO AP compulsory input*
InsertElement	ElementSetName
UpdateMetadata	Constraint
UpdateMetadataElement	RecordProperty
DeleteMetadata	Constraint
MetadataIdList	Id

Table 10 - Mapping between INSPIRE PublishMetadata input parameters and OGC Transaction important ones

The following table aims to explain in a more detailed way some of the OGC used input parameters:

Name	*Data type and value*	*Multiplicity and use*	*ISO Metadata Profile*
ELEMENTSETNAME	**CodeList** with allowed values: “brief”, “summary” or “full”	*Zero or one (Optional)*	*Zero or one (Optional)* Default value is “summary”
ID	**Comma separated list** of *anyURI*	*One (Mandatory)*	*One (Mandatory)*

Table 11 - OGC Transaction specific input parameter description

In the following listing, we report the OGC *TransactionRequest* message structure:

```
<wsdl:message name="TransactionRequest">
  <wsdl:part name="Body" element="csw:Transaction"/>
</wsdl:message>
```

```
<xsd:element name="Transaction" type="csw:TransactionType" id="Transaction"/>
 <xsd:complexType name="TransactionType" id="TransactionType">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
    Users may insert, update, or delete catalogue entries. If the verboseResponse attribute has the value "true",
    then one or more csw:InsertResult elements must be included in the response.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:extension base="csw:RequestBaseType">
    <xsd:sequence>
     <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="Insert" type="csw:InsertType"/>
      <xsd:element name="Update" type="csw:UpdateType"/>
      <xsd:element name="Delete" type="csw:DeleteType"/>
     </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="verboseResponse" type="xsd:boolean" use="optional" default="false"/>
    <xsd:attribute name="requestId" type="xsd:anyURI" use="optional"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

Listing 57 - OGC Transaction WSDL request message structure

5.1.1.4.2 OGC Response

In all the sub-operation cases (*insert*, *update* and *delete*), the main OGC *TransactionResponse* element is taken under consideration in INSPIRE, with different names in function of the sub-operation:

INSPIRE.PublishMetadata Output	*CSW ISO AP compulsory output*
InsertResult	TransactionResponse
UpdateResult	TransactionResponse
DeleteResult	TransactionResponse

Table 12 - Mapping between INSPIRE PublishMetadata output parameters and OGC transaction

The transaction response message conveys two pieces of information. First of all, it reports a summary of the transaction by indicating the number of records created, updated or deleted by the transaction. Secondly, the transaction response message indicates the results of each insert operation found in the transaction in the form of the <*InsertResult*> element.
The <*InsertResult*> element may appear zero or more times in the transaction response. It is used to report to the client a brief representation of each new record, including the record identifier, created in the catalogue. The records must be reported in the same order in which the <*Insert*> elements appear in a transaction request and must map 1 to 1.

```
<wsdl:message name="TransactionResponse">
  <wsdl:part element="csw:TransactionResponse" name="Body"/>
</wsdl:message>
```

```
<xsd:element name="TransactionResponse" type="csw:TransactionResponseType" id="TransactionResponse"/>
 <xsd:complexType name="TransactionResponseType">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
    The response for a transaction request that was successfully completed. If the transaction failed for any
    reason, a service exception report indicating a TransactionFailure is returned instead.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
   <xsd:element name="TransactionSummary" type="csw:TransactionSummaryType"/>
   <xsd:element name="InsertResult" type="csw:InsertResultType" minOccurs="0 maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:string" use="optional"/>
 </xsd:complexType>
```

Listing 58 - OGC Transaction WSDL response message structure

5.1.1.4.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data encoding, obtaining the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows=" http://www.opengis.net/ows/1.1"
  xmlns:ogc=" http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >

 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
   <s:element name="PublishMetadataRequest">
    <s:complexType>
     <s:sequence>
      <s:choice minOccurs="1" maxOccurs="unbounded">
        <s:element name="Insert" type="csw:InsertType" />
        <s:element name="Update" type="csw:UpdateType" />
        <s:element name="Delete" type="csw:DeleteType" />
      </s:choice>
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="PublishMetadataResponse" type="csw:TransactionResponseType" />
  </s:schema>
 </wsdl:types>

 <wsdl:message name="PublishMetadataSoapIn">
  <wsdl:part name="parameters" element="tns:PublishMetadataRequest" />
 </wsdl:message>
 <wsdl:message name="PublishMetadataSoapOut">
  <wsdl:part name="parameters" element="tns:PublishMetadataResponse" />
 </wsdl:message>

 <wsdl:portType name="DiscoveryServiceSoap">
  <wsdl:operation name="PublishMetadata">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The Publish Metadata operation allows
   to create, delete, or update (set) metadata (record) elements of spatial resources in the Discovery Service
   datastore.</wsdl:documentation>
   <wsdl:input message="tns:PublishMetadataSoapIn" />
   <wsdl:output message="tns:PublishMetadataSoapOut" />
   <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
  </wsdl:operation>
```

```
  </wsdl:portType>

  <wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
   <wsdl:documentation>
    <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
      xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
   </wsdl:documentation>
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
   <wsdl:operation name="PublishMetadata">
    <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/PublishMetadata" style="document" />
    <wsdl:input>
     <soap:body use="literal" />
    </wsdl:input>
    <wsdl:output>
     <soap:body use="literal" />
    </wsdl:output>
    <wsdl:fault name="ServiceExceptionReport">
      <soap:body use="literal" name="ServiceExceptionReport" />
    </wsdl:fault>
   </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

Listing 59 - PublishMetadata WSDL parts

5.1.1.4.4 SOAP Request Message

Here after we report some sample SOAP/1.1 request messages for the *PublishMetadata* method, one for each sub-operation:

Insert

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <PublishMetadataRequest xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
    <csw:Insert>
      <Record xmlns="http://www.opengis.net/cat/csw" xmlns:dc="http://www.purl.org/dc/elements/1.1/"
       xmlns:dct="http://www.purl.org/dc/terms/" xmlns:ows="http://www.opengis.net/ows" >
        …
      </Record>
    </csw:Insert>
  </PublishMetadata>
 </soap:Body>
</soap:Envelope>
```

Listing 60 - Sample PublishMetadata Insert SOAP request message

The <*record*> element has to be compliant with the Metadata IR, based on ISO 19139 encoding.

Update

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc=" http://www.opengis.net/ogc"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
 <soap:Body>
   <PublishMetadataRequest xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
     <csw:Update>
       <csw:RecordProperty>
         <csw:Name>/csw:Record/dc:contributor</csw:Name>
         <csw:Value>Jane</csw:Value>
       </csw:RecordProperty>
       <csw:Constraint>
         <ogc:Filter>
           <ogc:PropertyIsEqualTo>
             <ogc:PropertyName>/csw:Record/dc:contributor</ogc:PropertyName>
             <ogc:Literal>John</ogc:Literal>
           </ogc:PropertyIsEqualTo>
         </ogc:Filter>
       </csw:Constraint>
     </csw:Update>
   </PublishMetadata>
 </soap:Body>
```

```
</soap:Envelope>
```

Listing 61 - Sample PublishMetadata Update SOAP request message

Delete

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ogc=" http://www.opengis.net/ogc"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
 <soap:Body>
  <PublishMetadataRequest xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
    <csw:Delete>
      <csw:Constraint>
        <ogc:Filter>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>/csw:Record/dc:contributor</ogc:PropertyName>
            <ogc:Literal>Jane</ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:Filter>
      </csw:Constraint>
    </csw:Delete>
  </PublishMetadata>
 </soap:Body>
</soap:Envelope>
```

Listing 62 - Sample PublishMetadata Delete SOAP request message

5.1.1.4.5 SOAP Response Message

Here after we reproduced some sample SOAP/1.1 response messages for the *PublishMetadata* method, one for each sub-operation:

Insert

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
  <soap:Body>
    <PublishMetadataResponse>
      <csw:totalInserted>1</csw:totalInserted>
    </PublishMetadataResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 63 - Sample PublishMetadata Insert SOAP response message

Update

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
  <soap:Body>
    <PublishMetadataResponse>
      <csw:totalUpdated>1</csw:totalUpdated>
    </PublishMetadataResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 64 - Sample PublishMetadata Update SOAP response message

Delete

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >
  <soap:Body>
    <PublishMetadataResponse>
      <csw:totalDeleted>1</csw:totalDeleted>
    </PublishMetadataResponse>
  </soap:Body>
</soap:Envelope>
```

Listing 65 - Sample PublishMetadata Delete SOAP response message

5.1.1.5 CollectMetadata

The *CollectMetadata* operation allows to pull metadata (record) elements of spatial resources from a source *Discovery Service* data-store and allows to create, delete or update (set) the metadata (record) elements of these spatial resources in the target *Discovery Service* data-store (pull metadata mechanism).

The WSDL *portType* component of the *harvest* interface is shown in following Listing; this is a fragment of the complete WSDL 2.0 definition for the CSW Catalogue ([CSW XML-INT] and [CSW DISC]).

```
<wsdl:operation name="csw.harvest">
  <wsdl:input message="csw-req:HarvestRequest"/>
  <wsdl:output message="csw-resp:HarvestResponse"/>
  <wsdl:fault name="ServiceExceptionReport" message="csw-resp:ServiceExceptionReport"/>
</wsdl:operation>
```

```
<wsdl:operation name="csw.harvest">
  <soap:operation soapAction="http://www.opengis.net/cat/csw/2.0.2/requests#Harvest"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceExceptionReport">
    <soap:fault use="literal" name="ServiceExceptionReport"/>
  </wsdl:fault>
</wsdl:operation>
```

Listing 66 - WSDL CSW_Harvest interface

5.1.1.5.1 OGC Request

The *Harvest* operation has two modes of operation, controlled by a flag in the request. The first mode of operation is a synchronous mode in which the CSW receives a *Harvest* request from the client, processes it immediately and sends the results to the client while the client waits. The second mode is asynchronous in that the server receives a *Harvest* request from the client, and sends the client an immediate acknowledgement that the request has been successfully received (see [07-006]).

Here after we report the usual mapping between the CSW ISO AP input parameters and the INSPIRE useful ones:

INSPIRE.CollectMetadata Input	*CSW ISO AP compulsory input*
CollectSourceReference	Source
CollectMetadataResponseHandler	ResponseHandler
CollectInterval	HarvastInterval

Table 13 - Mapping between INSPIRE CollectMetadata input parameters and OGC Harvest

In the following listing, we report the OGC *HarvestRequest* message structure:

```
<wsdl:message name="HarvestRequest">
  <wsdl:part name="Body" element="csw:Harvest"/>
</wsdl:message>
```

```
<xsd:element name="Harvest" type="csw:HarvestType" id="Harvest"/>
 <xsd:complexType name="HarvestType">
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
   Requests that the catalogue attempt to harvest a resource from some network location identified by the
   source URL.
   Source - a URL from which the resource is retrieved
   ResourceType - normally a URI that specifies the type of the resource (DCMES v1.1) being harvested if it is
     known.
   ResourceFormat - a media type indicating the format of the resource being harvested.  The default is
     "application/xml".
   ResponseHandler - a reference to some endpoint to which the response shall be forwarded when the harvest
     operation has been completed
   HarvestInterval - an interval expressed using the ISO 8601 syntax; it specifies the interval between harvest
     attempts (e.g., P6M indicates an interval of six months).
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:extension base="csw:RequestBaseType">
    <xsd:sequence>
     <xsd:element name="Source" type="xsd:anyURI"/>
     <xsd:element name="ResourceType" type="xsd:string"/>
     <xsd:element name="ResourceFormat" type="xsd:string" minOccurs="0" default="application/xml"/>
     <xsd:element name="HarvestInterval" type="xsd:duration" minOccurs="0"/>
     <xsd:element name="ResponseHandler" type="xsd:anyURI" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

Listing 67 - OGC Harvest WSDL request message structure

5.1.1.5.2 OGC Response

In the following table we can see the mapping between the INSPIRE and the OGC main response parameter:

INSPIRE.CollectMetadata Output	*CSW ISO AP compulsory Output*
CollectResult	HarvestResponse

Table 14 - Mapping between INSPIRE CollectMetadata output parameter and OGC harvest one

This is the OGC method WSDL response message structure, together with the data and data types used:

```
<wsdl:message name="HarvestResponse">
  <wsdl:part element="csw:HarvestResponse" name="Body"/>
</wsdl:message>
```

```
<xsd:element name="HarvestResponse" type="csw:HarvestResponseType" id="HarvestResponse">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The content of the response varies depending on the presence of the ResponseHandler element. If present, then the catalogue should verify the request and respond immediately with an csw:Acknowledgement element in the response. The catalogue must then attempt to harvest the resource at some later time and send the response message to the location specified by the value of the ResponseHandler element using the indicated protocol (e.g. ftp, mailto, http).
      If the ResponseHandler element is absent, then the catalogue must attempt to harvest the resource immediately and include a TransactionResponse element in the response.
      In any case, if the harvest attempt is successful the response shall include summary representations of the newly created catalogue item(s).
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:complexType name="HarvestResponseType" id="HarvestResponseType">
  <xsd:choice>
    <xsd:element ref="csw:Acknowledgement"/>
    <xsd:element ref="csw:TransactionResponse"/>
  </xsd:choice>
</xsd:complexType>
```

Listing 68 - OGC Transaction WSDL response message structure

5.1.1.5.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data encoding, obtaining the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:ows=" http://www.opengis.net/ows/1.1"
 xmlns:ogc=" http://www.opengis.net/ogc" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/Discovery"
 xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" >

 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/Discovery">
   <s:element name="CollectMetadataRequest">
    <s:complexType>
     <s:sequence>
      <s:element name="Source" type="s:anyURI" />
      <s:element name="ResponseHandler" type="s:anyURI" />
      <s:element name="HarvestInterval" type="s:Duration" />
     </s:sequence>
    </s:complexType>
   </s:element>
   <s:element name="CollectMetadataResponse" type="csw:HarvestResponseType" />
  </s:schema>
 </wsdl:types>
 <wsdl:message name="CollectMetadataSoapIn">
  <wsdl:part name="parameters" element="tns:CollectMetadataRequest" />
 </wsdl:message>
 <wsdl:message name="CollectMetadataSoapOut">
  <wsdl:part name="parameters" element="tns:CollectMetadataResponse" />
 </wsdl:message>
 <wsdl:portType name="DiscoveryServiceSoap">
  <wsdl:operation name="CollectMetadata">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The Collect Metadata operation allows
    to pull metadata (record) elements of spatial resources from a source Discovery Service datastore and allows
    to create, delete or update (set) the metadata (record) elements of these spatial resources in the target
    Discovery Service datastore (pull metadata mechanism).</wsdl:documentation>
   <wsdl:input message="tns:CollectMetadataSoapIn" />
   <wsdl:output message="tns:CollectMetadataSoapOut" />
   <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
  </wsdl:operation>
 </wsdl:portType>

 <wsdl:binding name="DiscoveryServiceSoap" type="tns:DiscoveryServiceSoap">
```

```
    <wsdl:documentation>
      <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
        xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
    </wsdl:documentation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="CollectMetadata">
      <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/Discovery/CollectMetadata" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="ServiceExceptionReport">
          <soap:body use="literal" name="ServiceExceptionReport" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

</wsdl:definitions>
```

Listing 69 - CollectMetadata WSDL parts

5.1.1.5.4 SOAP Request Message

Here after we produced a sample SOAP/1.1 request message for the *CollectMetadata* method, one for each sub-operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:csw=”http://www.opengis.net/cat/csw/2.0.2” >
   <soap:Body>
      <CollectMetadata xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
         <csw:Source>http://www.myhost.com/metadata_dataset.xml</csw:Source>
         <csw:ResourceType>http://www.isotc211.org/schemas/2005/gmd/</csw:ResourceType>
         <csw:ResourceFormat>application/xml</csw:ResourceFormat>
         <csw:HarvestInterval>P1Y2M3DT10H30M0S</csw:HarvestInterval>
         <csw:ResponseHandler>ftp://ftp.myserver.com/HarvestResponses</csw:ResponseHandler>
      </CollectMetadata>
   </soap:Body>
</soap:Envelope>
```

Listing 70 - Sample CollectMetadata SOAP request message

5.1.1.5.5 SOAP Response Message

Here after we produced a sample SOAP/1.1 response message for the *CollectMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:csw=”http://www.opengis.net/cat/csw/2.0.2” >
 <soap:Body>
   <CollectMetadataResponse xmlns="http://inspire.jrc.ec.europa.eu/Discovery">
      <csw:Acknowledgement>
         <csw:EchoedRequest>….</csw:EchoedRequest>
         <csw:RequestId>….</csw:RequestId>
      </csw:Acknowledgement>
   </CollectMetadataResponse>
 </soap:Body>
</soap:Envelope>
```

Listing 71 - Sample CollectMetadata SOAP response message

5.1.2 Attachments

No attachments are foreseen for any of the INSPIRE *Discovery Service* operations.

5.2 View Service

In this section we are going to follow the same way undertaken for *Discovery Service*, illustrating the single methods that compose this INSPIRE service, going then in depth into each one of those, looking at the mapping between OGC parameters and INSPIRE ones and consequently how the WSDL fragments for the single operations are generated.

An INSPIRE *View Service* is a Web service which allows requests over geo-referenced data belonging to the themes covered by the INSPIRE Directive Annexes, over a spatial-temporal extension, and provides a visual representation of these data.

5.2.1 INSPIRE View Service operations

The *Network Services Drafting Team* made a proposal for the technical content of the INSPIRE Implementing Rules for Discovery Service; such proposal is available and described in "Draft Technical Guidance View Services v0.2" [INSPIRE TechG View/0.2].
Hereafter we report the mapping between the operations proposed by [OGC WMS] and the INSPIRE *View Service* operations (in the cardinality column, 'M' stands for Mandatory, while 'O' stands for Optional):

INSPIRE View Services functions	*INSPIRE Cardinality*	*OGC WMS operations*
INSPIRE.GetServiceMetadata	M	OGC_Service.GetCapabilities
INSPIRE.GetMap	M	WMS GetMap
INSPIRE.GetFeatureInformation	O	WMS GetFeatureInfo

Table 15 - INSPIRE View service and OGC WMS operations mapping

These three operations shall use parameters defined in the ISO 19128 WMS standard.

Accordingly with the INSPIRE SOAP Framework [EUR 23635 - 2008] all operations will support the embedding of requests and responses in SOAP 1.1 messages with Document/literal wrapped style.

5.2.1.1 GetServiceMetadata

According to Article 11, the metadata of a *View Service* shall be available through the *GetServiceMetadata* operation and even through the WMS service capabilities.
These metadata consist of server's information, supported operations, parameters values and dataset metadata URL.
The purpose of the mandatory *GetCapabilities* operation is to obtain service metadata, which is a machine-readable (and human-readable) description of the server's information content and acceptable request parameter values.

The WSDL binding component of WMS about this operation is shown in the following Listing; this fragment is reported from [04-050r1].

```
<wsdl:binding name="WMS_SOAP_Binding" type="wms-req:WMS_XML_Port">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetCapabilities">
      <soap:operation/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="exception">
        <soap:fault name="exception" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

Listing 72 - WSDL WMS getCapabilities interface

5.2.1.1.1 OGC Request

In the following table it's possible to see the *GetServiceMetadata* input parameters, their description and whether they are mandatory or not.

Request parameter	*Mandatory Optional*	*Description*
VERSION=version	O	Request version
SERVICE="WMS"	M	Service type
REQUEST="GetCapabilities"**	M	Request name
FORMAT=*MIME_type*	O	Output format of service metadata
UPDATESEQUENCE=string	O	Sequence number or string for cache control
LANGUAGE=code*	M	Request language

Table 16 – WMS GetCapabilities parameters

* In INSPIRE SOAP implementation, the *Language* parameter will be not taken under consideration, because, as explained in the proposed INSPIRE SOAP framework, the multilingual aspect will be managed exploiting a specific SOAP header (see *§4.8.4, Metadata & Piggy-backing - Multilingualism*).

** The *request* parameter won't be used in the INSPIRE *GetServiceMetadata* method, following what already specified in *owsGetCapabilities* XML Schema Definition: "In this XML encoding, no *request* parameter is included, since the element name (*getCapabilities*) specifies the specific operation".

OGC does not provide a *GetCapabilities* WSDL request message structure.

5.2.1.1.2 OGC Response

When invoked on a *Web Map Service*, the response to a *GetCapabilities* request shall be an XML document containing service metadata.
There are two parts in the *Capabilities* of a *View Service*: the first one is related to the service itself, the second one is related to the layers this service is able to visualise.

OGC does not provide a *GetCapabilities* WSDL response message structure, but supplies an XML schema for the response data type (*WMS_Capabilities*):

```
<element name="WMS_Capabilities">
  <annotation>
    <documentation>
      A WMS_Capabilities document is returned in response to a GetCapabilities request made on a WMS.
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element ref="wms:Service"/>
      <element ref="wms:Capability"/>
    </sequence>
    <attribute name="version" type="ows:VersionType" use="required"/>
    <attribute name="updateSequence" type="string"/>
  </complexType>
</element>
```

Listing 73 - OGC GetCapabilities response data and data types

5.2.1.1.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data style and encoding to define the WSDL morphology and so, for this operation, we obtain the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://inspire.jrc.ec.europa.eu/View"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://inspire.jrc.ec.europa.eu/View" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wms="http://www.opengis.net/wms" >

 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/View">
   <s:element name="GetCapabilities">
    <s:complexType>
     <s:sequence>
       <s:element minOccurs="0" ref="wms:Format"/>
     </s:sequence>
     <s:attribute use="optional" name="version" type="ows:VersionType"/>
     <s:attribute use="required" name="service" type="s:string"/>
     <s:attribute use="optional" name="updateSequence" type="s:string"/>
    </s:complexType>
   </s:element>
  </s:schema>
 </wsdl:types>

 <wsdl:message name="GetServiceMetadataSoapIn">
  <wsdl:part name="parameters" element="tns:GetCapabilities" />
 </wsdl:message>
 <wsdl:message name="GetServiceMetadataSoapOut">
  <wsdl:part name="parameters" element="wms:WMS_Capabilities" />
 </wsdl:message>

 <wsdl:portType name="ViewServiceSoap">
  <wsdl:operation name="GetServiceMetadata">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The goal of this method is to obtain
     service metadata, which is a machine-readable (and human-readable) description of the server's information
     content and acceptable request parameter values
   </wsdl:documentation>
   <wsdl:input message="tns:GetServiceMetadataSoapIn" />
   <wsdl:output message="tns:GetServiceMetadataSoapOut" />
   <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
  </wsdl:operation>
 </wsdl:portType>
```

```
 <wsdl:binding name="ViewServiceSoap" type="tns:ViewServiceSoap">
  <wsdl:documentation>
   <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
    xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
  </wsdl:documentation>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetServiceMetadata">
   <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/View/GetServiceMetadata" style="document" />
   <wsdl:input>
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
   <wsdl:fault name="ServiceExceptionReport">
     <soap:body use="literal" name="ServiceExceptionReport" />
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>

</wsdl:definitions>
```

Listing 74 - GetDiscoveryServiceMetadata WSDL parts

5.2.1.1.4 SOAP Request Message

Here after we report a sample SOAP/1.1 request message for the *GetServiceMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
   <GetCapabilities xmlns="http://inspire.jrc.ec.europa.eu/View" service="WMS" version="1.3.0" />
  </soap:Body>
</soap:Envelope>
```

Listing 75 - Sample GetDiscoveryServiceMetadata SOAP request message

5.2.1.1.5 SOAP Response message

Here after we reproduced a sample SOAP/1.1 response message for the *GetServiceMetadata* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
    <wms:Capabilities version="1.3.0" …. />
 </soap:Body>
</soap:Envelope>
```

Listing 76 - Sample GetDiscoveryServiceMetadata SOAP response message

5.2.1.2 GetMap

The *GetMap* operation returns a map. Upon receiving a *GetMap* request, a WMS shall either satisfy the request or issue a service exception.

The WSDL binding component of WMS about this operation is shown in the following Listing; this fragment is reported from [04-050r1].

```
<wsdl:binding name="WMS_SOAP_Binding" type="wms-req:WMS_XML_Port">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetMap">
      <soap:operation/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <mime:content type="image/*"/>
      </wsdl:output>
      <wsdl:fault name="exception">
        <soap:fault name="exception" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

Listing 77 - WSDL WMS getMap interface

5.2.1.2.1 OGC Request

In the following table it's possible to see the *GetMap* input parameters, their description and whether they are mandatory or not.

Request parameter	*Mandatory Optional*	*Description*
VERSION	M	Request version
REQUEST = "GetMap"**	M	Request name
LAYERS = layer_list	M	Comma-separated list of one or more map layers.
STYLES = style_list	M	Comma-separated list of one rendering style per requested
CRS = namespace:identifier	M	Coordinate reference system
BBOX = minx,miny,maxx,maxy	M	Bounding box corners (lower left, upper right) in CRS units
WIDTH	M	Width in pixels of map picture
HEIGHT	M	Height in pixels of map picture
FORMAT	M	Output format of map. At least supported : Portable Network Graphics format(PNG; MIME type "image/png") and the GIF (Graphics Interchange Format) without LZW compression (MIME type "image/gif")
LANGUAGE*	M	Request language
TRANSPARENT = TRUE|FALSE	O	Background transparency of map (default=FALSE)

BGCOLOR = *color_value*	O	Hexadecimal red-green-blue color value for the background color (default=0xFFFFFF)
EXCEPTIONS = *exception_format*	O	The format in which exceptions are to be reported by the WMS (default=XML)
TIME	O	Time value of layer desired
ELEVATION	O	Elevation of layer desired
Other sample dimension(s)	O	Value of other dimensions as appropriate

Table 17 – WMS GetCapabilities parameters

* In INSPIRE SOAP implementation, the *Language* parameter will be not taken under consideration, because, as explained in the proposed INSPIRE SOAP framework, the multilingual aspect will be managed exploiting a specific SOAP header.
** The *request* parameter won't be used in the INSPIRE *GetMap* method, following what already specified in owsGetCapabilities XML Schema Definition: "In this XML encoding, no *request* parameter is included, since the element name (*getMap*) specifies the specific operation".

OGC does not provide a *GetCapabilities* WSDL request message structure, neither a specific XML schema for the request data types.

5.2.1.2.2 OGC Response

The response to a valid *GetMap* request shall be a map of the spatially referenced information layer requested, in the desired style and having the specified coordinate reference system, bounding box, size, format and transparency.
An invalid *GetMap* request shall yield an error output in the requested exceptions format (or a network protocol error response in extreme cases).
In an HTTP environment, the MIME type of the returned value's Content-type entity header shall match the format of the return value.

OGC does not provide a *GetMap* WSDL response message structure and neither supplies an XML schema for the response data type.

5.2.1.2.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data style and encoding to define the WSDL morphology and so, for this operation, we obtain the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/View"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://inspire.jrc.ec.europa.eu/View" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wms="http://www.opengis.net/wms" >
 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/View">
   <s:element name="GetMapRequest">
    <s:complexType>
     <s:sequence>
      <s:element name="layers" type="s:string"/>
      <s:element name="styles" type="s:string"/>
      <s:element ref="wms:CRS"/>
      <s:element ref="wms:BoundingBox"/>
      <s:element name="width" type="s:nonNegativeInteger"/>
      <s:element name="height" type="s:nonNegativeInteger"/>
      <s:element ref="wms:Format"/>
      <s:element minOccurs="0" name="transparent" type="s:boolean"/>
      <s:element minOccurs="0" name="bgcolor" type="s:string"/>
      <s:element minOccurs="0" ref="wms:Format"/>
      <s:element minOccurs="0" name="time" type="s:dateTime"/>
      <s:element minOccurs="0" name="elevation" type="s:string"/>
      <s:element minOccurs="0" ref="wms:Dimension"/>
     </s:sequence>
     <s:attribute name="version" type="ows:VersionType"/>
    </s:complexType>
   </s:element>
   <s:element name="GetMapResponse" type="s:base64Binary" />
  </s:schema>
 </wsdl:types>

 <wsdl:message name="GetMapSoapIn">
  <wsdl:part name="parameters" element="tns:GetMapRequest" />
 </wsdl:message>
 <wsdl:message name="GetMapSoapOut">
  <wsdl:part name="parameters" element="tns:GetMapResponse" />
 </wsdl:message>

 <wsdl:portType name="ViewServiceSoap">
```

```
  <wsdl:operation name="GetMap">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The goal of this method is to return a
     map</wsdl:documentation>
   <wsdl:input message="tns:GetMapSoapIn" />
   <wsdl:output message="tns:GetMapSoapOut" />
   <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
  </wsdl:operation>
 </wsdl:portType>

 <wsdl:binding name="ViewServiceSoap" type="tns:ViewServiceSoap">
  <wsdl:documentation>
   <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
    xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
  </wsdl:documentation>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetMap">
   <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/View/GetMap" style="document" />
   <wsdl:input>
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
   <wsdl:fault name="ServiceExceptionReport">
     <soap:body use="literal" name="ServiceExceptionReport" />
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>

</wsdl:definitions>
```

Listing 78 - GetMap WSDL parts

5.2.1.2.4 SOAP Request Message

Here after we report a sample SOAP/1.1 request message for the *GetMap* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <GetMap xmlns="http://inspire.jrc.ec.europa.eu/View" version="1.3.0" service="WMS"
       xmlns:gml="http://www.opengis.net/gml" >
         <Layers>highways</Layers>
         <Styles>ROADS_RIVERS</Styles>
         <CRS>EPSG:4326</CRS>
         <bbox CRS="EPSG:4326" minx=-71.08503056559105 miny=42.29748471720447  maxx=71.07496943440891
            maxy=42.30251528279554</ bbox >
         <Format>image/png</Format>
         <Transparent>true</Transparent>
         <BGcolor>0xFFFFFF</BGcolor>
         <Width>400</Width>
         <Height>200</Height>
         <Exceptions>application/vnd.ogc.se_xml</Exceptions>
      </GetMap>
   </soap:Body>
</soap:Envelope>
```

Listing 79 - GetMap SOAP request message

5.2.1.2.5 SOAP Response message

Here after we reproduced a sample SOAP/1.1 response message for the *GetMap* operation:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
   <GetMapResponse xmlns="http://inspire.jrc.ec.europa.eu/View">
      /RTRfkgnflkgnf=fndslkfnkldsnfkldsnf43d5fwfsgs4ojojsfiodjs…=
   </GetMapResponse>
 </soap:Body>
</soap:Envelope>
```

Listing 80 - Sample GetMap SOAP response message

5.2.1.3 *GetFeatureInformation*

GetFeatureInfo is an optional operation. It is only supported for those Layers for which the attribute *queryable="1"* (true) has been defined or inherited. A client shall not issue a *GetFeatureInfo* request for other layers. A WMS shall respond with a properly formatted service exception (XML) response (code = *OperationNotSupported*) if it receives a *GetFeatureInfo* request but does not support it.
The *GetFeatureInfo* operation is designed to provide clients of a WMS with more information about features in the pictures of maps that were returned by previous *GetMap* requests. The canonical use case for *GetFeatureInfo* is that a user sees the response of a *GetMap* request and chooses a point (I,J) on that map for which to obtain more information. The basic operation provides the ability for a client to specify which pixel is being asked about, which layer(s) should be investigated, and what format the information should be returned in. Because the WMS protocol is stateless, the *GetFeatureInfo* request indicates to the WMS what map the user is viewing by including most of the original *GetMap* request parameters (all but VERSION and REQUEST).
From the spatial context information (BBOX, CRS, WIDTH, HEIGHT) in that *GetMap* request, along with the I,J position the user chose, the WMS can (possibly) return additional information about that position.
There are no WSDL binding components of WMS about this optional method.

5.2.1.3.1 OGC Request

In the following table it's possible to see the *GetFeatureInfo* input parameters, their description and whether they are mandatory or not.

Request parameter	*Mandatory Optional*	*Description*
VERSION	M	Request version
REQUEST = "GetFeatureInfo"*	M	Request name
map request part	M	Partial copy of the Map request parameters that generated the map for which information is desired
QUERY_LAYERS = layer_list	M	Comma-separated list of one or more layers to be queried
INFO_FORMAT =output_format	M	Return format of feature information (MIME type).
FEATURE_COUNT = number	O	Number of features about which to return information (default=1).
I	M	i coordinate in pixels of feature in Map CS
J	M	j coordinate in pixels of feature in Map CS
EXCEPTIONS =exception_format	O	The format in which exceptions are to be reported by the WMS (default= XML).

Table 18 – WMS GetFeatureInformation parameters

* The *request* parameter won't be used in the INSPIRE *GetFeatureInformation* method, following what already specified in *owsGetCapabilities* XML Schema Definition: "In this XML encoding, no *request* parameter is included, since the element name (*getFeatureInformation*) specifies the specific operation".

OGC does not provide a *GetFeatureInfo* WSDL request message structure.

5.2.1.3.2 OGC Response

The *Web Map Server* shall return a response according to the requested INFO_FORMAT if the request is valid, or issue a service exception otherwise. The nature of the response is at the discretion of the service provider, but it shall pertain to the feature(s) nearest to (I,J).

OGC does not provide a *GetFeatureInfo* WSDL response message structure and neither supplies an XML schema for the response data type.

5.2.1.3.3 INSPIRE Operation WSDL

In this paragraph we exemplify the definition of the *GetDiscoveryServiceMetadata* operation, accordingly with the INSPIRE SOAP framework [EUR 23635 - 2008], by relying on the OGC CSW specification as indicated in [INSPIRE TechG Disc/1.0].
We will use a **Document-literal wrapped** data style and encoding to define the WSDL morphology and so, for this operation, we obtain the following sample WSDL schema:

```
<?xml version="1.0" encoding="utf-16"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://inspire.jrc.ec.europa.eu/View"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  targetNamespace="http://inspire.jrc.ec.europa.eu/View" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

 <wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://inspire.jrc.ec.europa.eu/View">
   <s:element name="GetFeatureInformation">
    <s:complexType>
     <s:sequence>
      <s:element name="mapRequestPart" type="s:string"/>
      <s:element name="query_layers" type="s:string"/>
      <s:element ref="wms:Format"/>
      <s:element name="i" type="s:nonNegativeInteger"/>
      <s:element name="j" type="s:nonNegativeInteger"/>
      <s:element minOccurs="0" maxOccurs="1" name="feature_count" type="s:nonNegativeInteger"/>
      <s:element minOccurs="0" maxOccurs="1" ref="wms:Format"/>
     </s:sequence>
     <s:attribute name="version" type="ows:VersionType" use="required"/>
    </s:complexType>
   </s:element>
   <s:element name="GetFeatureInformationResponse" type="s:base64Binary" />
  </s:schema>
 </wsdl:types>

 <wsdl:message name="GetFeatureInformationSoapIn">
  <wsdl:part name="parameters" element="tns:GetFeatureInformation" />
 </wsdl:message>
 <wsdl:message name="GetFeatureInformationSoapOut">
  <wsdl:part name="parameters" element="tns:GetFeatureInformationResponse" />
 </wsdl:message>

 <wsdl:portType name="ViewServiceSoap">
```

```
  <wsdl:operation name="GetFeatureInformation">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">The goal of this method is to provide
     clients with more information about features in the pictures of maps that were returned by previous Map
     requests</wsdl:documentation>
   <wsdl:input message="tns:GetFeatureInformationSoapIn" />
   <wsdl:output message="tns:GetFeatureInformationSoapOut" />
   <wsdl:fault name="ServiceExceptionReport" message="insp:ServiceExceptionReport" />
  </wsdl:operation>
 </wsdl:portType>

 <wsdl:binding name="ViewServiceSoap" type="tns:ViewServiceSoap">
  <wsdl:documentation>
   <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.1"
    xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
  </wsdl:documentation>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="GetFeatureInformation">
   <soap:operation soapAction="http://inspire.jrc.ec.europa.eu/View/GetFeatureInformation" style="document" />
   <wsdl:input>
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal" />
   </wsdl:output>
   <wsdl:fault name="ServiceExceptionReport">
     <soap:body use="literal" name="ServiceExceptionReport" />
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>

</wsdl:definitions>
```

Listing 81 - GetFeatureInformation WSDL parts

5.2.1.3.4 SOAP Request Message

Here after we report a sample SOAP/1.1 request message for the *GetFeatureInformation* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <GetFeatureInformation xmlns="http://inspire.jrc.ec.europa.eu/View" version="1.0">
     <mapRequestPart/>
     <query_layers>highways<query_layers>
     <info_format>image/png</info_format>
     <feature_count>0</feature_count>
     <i>21</i>
     <j>69</j>
     <Exceptions>application/vnd.ogc.se_xml</Exceptions>
  </GetFeatureInformation>
 </soap:Body>
</soap:Envelope>
```

Listing 82 - Sample GetFeatureInformation SOAP request message

5.2.1.3.5 SOAP Response message

Here after we reproduced a sample SOAP/1.1 response message for the *GetFeatureInformation* operation:

```
<?xml version="1.0" encoding="utf-16"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <GetFeatureInformationResponse xmlns="http://inspire.jrc.ec.europa.eu/View">
    …
  </GetFeatureInformationResponse>
 </soap:Body>
</soap:Envelope>
```

Listing 83 - Sample GetFeatureInformation SOAP response message

6. Compliance with the INSPIRE SOAP framework

The WSDLs created and explained in this document have been generated following the technical decisions taken in [EUR 23635 - 2008], born in function of the actual SOAP standards maturity, the INSPIRE domain and with a look at OGC services specifications.

In both the services WSDLs analysed, there are no aspects that do not collide with the theoretical conclusions outlined in the framework:

- SOAP headers have been studied and exemplified for security, data checks, description and multilingualism;
- binary data have been attached in SOAP messages using MTOM+XOP and the whole mechanism has been explained;
- exceptions have been managed as suggested by OGC in its SOAP change proposals, but converting the data structure to be compliant with [SOAP/1.1] specification and WS-I Basic Profile 1.2;
- SOAP 1.1 on HTTP has been used, keeping WS-I Basic Profile 1.2 compliance;
- SOAP messages have been composed using Document-literal wrapped style and encoding.

No problem has occurred writing down both View and Discovery services WSDLs, neither any incongruence, nor incompatibility issue.

7. Conclusions

In this document we have examined the WSDL documents generated for INSPIRE Discovery and View services (fully reported in [WSDL Disc-View]), splitting them into main sections, explaining the reasons why some choices have been undertaken and how these choices have reflected on the XML code.

In the first section of the document (*§4 - SOAP messages and WSDL for INSPIRE Discovery and View Service*), the general aspects of the WSDL have been investigated, following a structure similar to the one of the WSDL 2.0 primer [WSDL/2.0], writing down specific sections, for example, about headers, binary data transport, exceptions, target namespaces and so on.
The headers topic has been investigated in depth, starting from the suggestions specified in [EUR 23635 - 2008], proposing and giving a sample implementation of four different *ad hoc* headers for different purposes (see Section 4.8).

In section *5 – Discovery and View services WSDL in depth*, a different approach has been followed: the single Discovery and View services WSDLs have been examined independently under a functional point of view, analysing the single Web service methods. A comparison between them and the corresponding OGC ones has been made, looking at parameters and data types. For each one of the Web methods a sample SOAP request and response message has been provided.

The analysed WSDL code results are totally conforming to the INSPIRE SOAP framework described in [EUR 23635 - 2008].

The Discovery and View Technical guidance document ([INSPIRE TechG Disc/1.0] and [INSPIRE TechG View/0.2]) do not explicitly consider the use of attachments and related optimisation mechanisms, or the headers topics introduced in this document. The multilingual requirements, is approached in a different way rather than the way discussed in this primer. The IR and Technical guidance documents suggested using an additional single input parameter for each Web method, to specify the client's be-wished language, to minimize the number of assumptions on the Service Client. Even if this approach solves the multilingual problem, this deliverable proposes a SOAP optimized solution, exploiting an *ad hoc* SOAP header, while still compliant with the Implementing Rule.

Regarding binary data attachments in the *GetMap* operation, a specific approach with correlated data transport optimisation would help ensure good performance, considering the dimensions that detailed map images can reach nowadays.

8. Annex A – Terms, Definitions and Abbreviations

ACRONYM		*EXPLANATION*
NSDT		Network Service Drafting Team
XSD		XML Schema Definition

Table 19 – Document Abbreviations Table

9. Annex B – Inspire IR Reference

REF.	*AUTHOR(S)*	*TITLE*	*DATE*
2007/2/EC		INSPIRE Directive	2007
INSPIRE NSA/3.0	NSDT	INSPIRE Network Services Architecture v3.0 http://inspire.jrc.ec.europa.eu/reports.cfm	2008

Table 20 - Inspire IR References

10. Annex C – Bibliography

The following table lists the sources referenced in the present document.

REF.	AUTHOR(S)	TITLE	DATE
04-050r1	Duschene P. Sonnet J.	OGC WMS Change Request: Support for WSDL & SOAP	2005
06-121r3	Whiteside A.	OGC Web Services Common Specification 1.1.0 http://www.opengeospatial.org/standards/	2007
07-006	Nebert D. Whiteside A. Vretanos P.	OGC Catalogue Services Specification http://www.opengeospatial.org/standards/	2007
CSW DISC	//	CSW-publication.xsd *(http://schemas.opengis.net/csw/2.0.2/CSW-publication.xsd)*	//
CSW PUB	//	CSW-discovery.xsd *(http://schemas.opengis.net/csw/2.0.2/CSW-discovery.xsd)*	//
CSW XML-INT	//	XML-interfaces.wsdl (*http://schemas.opengis.net/csw/2.0.2/examples/wsdl/2.0.2/xml-interfaces.wsdl*)	//
Ghosh, 2007	Ghosh V.	Transferring Data securely using the MTOM Standard	2007
ISO 639-2/B	TC37/SC2-TC46/SC4 Joint Working Group (JWG)	ISO 639-2: Codes for the representation of names of languages	2006
OGC CSW	Voges U. Senkler K.	OGC Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile http://www.opengeospatial.org/standards/cat	2007
OGC WMS	J. de la Beaujardiere	OGC Web Map Server Implementation Specification http://www.opengeospatial.org/standards/	2006
WSDL/2.0	Chinnici R., Moreau J. J. et al.	Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language http://www.w3.org/	2007

Table 21 - Bibliography

European Commission

EUR 23704 EN – Joint Research Centre – Institute for Environment and Sustainability
Title: SOAP Primer for INSPIRE Discovery and View Services
Author(s): Matteo Villa, Giovanni Di Matteo, Roberto Lucchi, Michel Millot, Ioannis Kanellopoulos
Luxembourg: Office for Official Publications of the European Communities
2009 – 96pp. – 21 x 29.7 cm
EUR – Scientific and Technical Research series – ISSN 1018-5593
ISBN 978-92-79-11317-8
DOI 10.2788/78376

Abstract
This document demonstrates the use of the proposed INSPIRE SOAP Framework for the INSPIRE Discovery and View services. This document focuses on the analysis of the WSDL itself (for both the Discovery and View services), explaining its parts and characteristics, as well as on the analysis of SOAP request and response messages, including headers and potential attachments. Moreover, the primer is providing also examples of user scenarios, with specific code samples.

How to obtain EU publications

Our priced publications are available from EU Bookshop (http://bookshop.europa.eu), where you can place an order with the sales agent of your choice.

The Publications Office has a worldwide network of sales agents. You can obtain their contact details by sending a fax to (352) 29 29-42758.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

LB-NA-23704-EN-C